THE BABY PROJECT:

PROCESSING CHARACTER PATTERNS IN TEXTUAL REPRESENTATIONS OF LANGUAGE

PAUL ANTON PETER ROGERS

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of Doctor of Philosophy

April 2000

Bournemouth University

Abstract.

The Baby Project: processing character patterns in textual representations of language.

Paul Anton Peter Rogers.

This thesis describes an investigation into a proposed theory of AI. The theory postulates that a machine can be programmed to predict aspects of human behaviour by selecting and processing stored, concrete examples of previously experienced patterns of behaviour. Validity is tested in the domain of natural language. Externalisations that model the resulting theory of NLP entail fuzzy components. Fuzzy formalisms may exhibit inaccuracy and/or overproductivity. A research strategy is developed, designed to investigate this aspect of the theory. The strategy includes two experimental hypotheses designed to test, 1) whether the model can process simple language interaction, and 2) the effect of fuzzy processes on such language interaction. Experimental design requires three implementations, each with progressive degrees of fuzziness in their processes. They are respectively named: NonfuzzBabe, CorrBabe and FuzzBabe. NonfuzzBabe is used to test the first hypothesis and all three implementations are used to test the second hypothesis. A system description is presented for NonfuzzBabe. Testing the first hypothesis provides results that show NonfuzzBabe is able to process simple language interaction. A system description for CorrBabe and FuzzBabe is presented. Testing the second hypothesis, provides results that show a positive correlation between degree of fuzzy processes and improved simple language performance. FuzzBabe's ability to process more complex language interaction is then investigated and model-intrinsic limitations are found. Research to overcome this problem is designed to illustrate the potential of externalisations of the theory and is conducted less rigorously than previous part of this investigation. Augmenting FuzzBabe to include fuzzy evaluation of non-pattern elements of interaction is hypothesised as a possible solution. The term FuzzyBaby was coined for augmented implementation. Results of a pilot study designed to measure FuzzyBaby's reading comprehension are given. Little research has been conducted that investigates NLP by the fuzzy processing of concrete patterns in language. Consequently, it is proposed that this research contributes to the intellectual disciplines of NLP and AI in general.

List of Contents.

List of Tables.

List of Figures.

PAGE
MISSING
IN
ORIGINAL

Accompanying material.

During the course of this research, papers were presented at various refereed conferences as follows:

Lefley, M. & P. Rogers (1997). Using Patterns in State Spaces of Complex Problems to Hypothesise Solution Behaviour. In Proceedings of: *Eighth Ireland Conference on Artificial Intelligence (AI-97),* Dublin.

Lefley, M. & P. Rogers (1998). Our Fuzzy Baby: using fuzzy competitive logic to learn natural language interpretation. In Proceedings of: *International Symposium on Engineering of Intelligent Systems EIS98,* Tenerife, Spain .

Lefley, M. & P. Rogers (1998a). Baby: a system that learns natural language processing by using pattern recognition and fuzzy competitive mechanisms. In Proceedings of: *Recent Advances in Soft Computing,* Leicester, UK. .

Lefley, M. & P. Rogers (1999). Incorporating Fuzzy Resonance in Language Acquisition and Production. In Proceedings of: *CIMA,* New York .

Rogers, P. & M. Lefley (1997). The Baby Project: an alternative approach to machine learning of wide domain natural language environments. In Proceedings of: *First International Workshop on Human-Computer Conversation,* Bellagio.

One paper was accepted for publication as a chapter in a book, as follows:

Rogers, P. & M. Lefley (1999). The Baby Project: an alternative approach to machine learning of wide domain natural language environments. In (ed. Y.A. Wilks): Machine Conversations., Boston, Kluwer Academic Press.

One paper was published in a refereed journal, as follows:

Rogers, P. & M. Lefley (1997a). "Processing Patterns in Text: an alternative approach to natural language processing." The European Students Journal of Language and Speech., Vol. 1, (#97/01).

Acknowledgements.

Special and deep thanks go to my supervisor and mentor, Dr. Martin Lefley, for his support and guidance, whilst giving me the freedom to follow aspects of my own research interests in this project.

I would also like to thank my external supervisor, Dr. Paul McKevitt and my secondary internal supervisors, Dr. Gerry Griffin and Alan Potton for their invaluable advice.

Thanks also to Prof. Brian MacWhinney for allowing me to use his corpus of English verbs.

I would also like to express my gratitude to all the members of the Speech and Language Interaction Group (SPLINTER) at Bournemouth University, but especially to Dan Vine and Jenny Longster. Thanks to Becky, my office mate, for her support through the bad times, laughs in the good and being there in the ordinary. Cheers to Ezzi, Ben and Charlie for their support and especially for taking me past the pig farm so profitably.

Finally, I would like to acknowledge the role of Bournemouth University, and the management in the Department of Design Engineering and Computing, especially to David Knight, Prof. Peter Hogarth and Prof. Martin Shepperd, for enabling the funding of this blue sky research and allowing me to act autonomously in pursuit of my research objectives.

1.0 Introduction.

This thesis describes part of a full time, uncollaborated research programme. The work comprises an investigation into a notion that a machine can be programmed to predict aspects of human behaviour by selecting and processing stored, concrete examples of previously experienced patterns of behaviour. For this research, human language behaviour is used to test the notion. The chapter begins with a brief account of the context of the discipline that subsumes this work. An overview of the research described in this thesis is given. The more important issues that motivate the work are then listed. This is followed by a statement of research goals. A list of expected constraints and limitations connected with the research is then given. Finally, a summary of the issues covered in the chapter is provided.

1.1 Context of research.

The domain Artificial Intelligence (AI) subsumes the research described in this thesis. Research into creating intelligent machines has generally been conducted within the parameters of two paradigms. The first, known as the symbolic approach, takes its underlying assumptions from the *physical symbol system*, first proposed by Newell & Simon (1976). Here, modular units represent a hierarchy of increasing levels of cognitive complexity. Beginning at an input or biological level, information is processed through a functional level that feeds a semantic representation level. Instantiations that model this approach contain processes where:

- knowledge is organised in discrete symbolic structures;
- the interaction of these symbolic structures is determined by serial control;
- the structures and control build a symbolic episodic memory.

The second paradigm, known as the connectionist approach, assumes the *connectionist system hypothesis* (Wermter 1995) and employs a non-hierarchical distributed representation of cognitive behaviour. Instantiations that model this approach contain processes where:

- knowledge is organised continuously across a network;
- the interaction between the nodes of the network is determined by parallel control;
- knowledge structures and control produce distributed episodic memory.

The research described in this thesis is more closely allied to the latter paradigm.

Early expectations of being able to program computers to simulate complex aspects of people's behaviour were high, (Manaris & Slator, 1996). It was thought that anything that a child of three could accomplish, such as catching a ball, understanding a story or seeing, was going to be relatively simple for a computer. This perception could have been influenced by the fact that computers could already accomplish tasks that few adults could achieve, such as complicated mathematical calculations at super human speeds. In retrospect, expectations were predictably optimistic. A child inherits a brain, that may be considered as a computing machine, whose operational parameters are so vast and whose interconnection so complex, as to defy understanding, (see Fischbach (1993) for a good account). To suppose that significant aspects of the behaviour of the human brain, that emergent jewel of an aeon of evolution, could be simulated by a relatively small assembly of valves, resistors, capacitors, relays and lamps etc. seems now, in the light of knowledge gained in the trying, difficult to believe. Many important lessons have been identified from the exercise. The method of simulating aspects of human behaviour employed in this research learns from those lessons and may avoid some historical pitfalls. It is hypothesised that this will offer the possibility of an alternative and exciting way of approaching the problem.

The theory of AI that generally underpins this research may be formally stated as follows:

> an element of the state of affairs of an intelligent conception of the world, is explicitly modelled in the set of concrete representations of patterns of behaviour that have been experienced and stored by the intelligence. Further, that an element of the state engendered by a new experience, is synthesised from the response component of those stored patterns of behaviour whose stimulus component most closely correspond with the pattern of the new experience.

The theory is tested in this research by the processing of natural language represented in the uni-modal sub-domain of text. Consequently, the theory that directly underpins this research may be stated as:

> for a given user of a language, a textual representation of an element of the response state engendered by experiencing a current textual stimulus, can be synthesised from textual response patterns associated with previously experienced stimuli, whose textual patterns most closely correspond with the textual patterns of the current textual stimulus.

This underpinning theory places the work close to a new approach to Natural Language Processing (NLP), that of Data Orientated Language Processing (DOLP). This approach was first posited by Scha (1990). Essential elements of the theory underpinning this research are shared with assumptions made for DOLP research. However, association of this work with DOLP may only be considered under a tenuous interpretation of some of DOLP's tenets (see section *2.8 Related research.* for more detail).

Approaching the problem of creating an artificial intelligence by allowing a machine to interact within the domain of simple language is not a new one, for instance, Turing, (1950) wrote:

> *We may hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess, would be best. It can also be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named, etc. Again, I do not know what the right answer is, but I think both should be tried.*

The research described here may be considered as the beginnings of an attempt to tackle the latter solution strategy, in the limited, uni-modal domain of textual representations of language.

1.2 Brief description of research.

The name coined for this project was **Baby.** This name was chosen for two reasons. Firstly, because externalisations of the underpinning theory were required to process language following small exposure to past language experience. Secondly, the suitability of the name was reinforced during the research, when a correlation was observed between such externalisations and children's language acquisition behaviour.

The Baby project entails background research, evolving a model of the underpinning theory, researching the design, implementation, test and performance measurement of externalisation of the model. Because the words 'most closely correspond' appear in the underlying theory, models of the theory employ fuzzy problem solving techniques to process language. Fuzzy formalisms may exhibit inaccuracy and overproductivity. In order to reduce the domain of this investigation, to one that was considered both significant and commensurate with prospective doctoral research, a strategy was developed that was designed to investigate this fuzzy problem solving aspect.

The strategy includes two experimental hypotheses with experimental set-ups that are designed to test:

1. whether the model can process simple language interaction;
2. the effect of fuzzy processes on such language interaction.

The experiment would require three implementations, each with progressive degrees of fuzziness in their processes. The implementation that possessed the least fuzziness in its processes would be used to test the first hypothesis and all three implementations would be used to test the second hypothesis. The ability of the models to process more complex language interaction would then be investigated. The investigation would be intended to illustrate the potential of externalisations of the theory and would therefore be conducted less rigorously than in previous parts of this investigation. Results of these various tests would be shown and analysed and conclusions that could be drawn would be stated.

1.3 Motivation.

There were four main motivations behind pursuing this project. Firstly, some, as then, unpublished research conducted by Dr. Martin Lefley of Bournemouth University, that the author found intellectually challenging. Secondly, the prospect that Dr. Lefley's research offered the possibility of circumventing some of the problems relating to machines that acquire the behaviour of complex problem spaces. Thirdly, to test the underpinning theory, that had been developed by the author over several years of both undergraduate and postgraduate research and that shared many of the basic principles behind Dr. Lefley's research. Fourthly, the test domain is recognised as being one of the most challenging problems within the AI community

1.4 Goals.

The following goals were set in order to test the extent to which this research finally produces significant results:

1. to indicate whether a low-fuzzy model of the underpinning theory is capable of processing simple language interaction, as judged by a human user of the language;
2. to indicate the extent to which the incorporation of fuzzy processes in the model, affect its simple language interaction performance, as judged by a human user of the language;
3. to measure the relative performance between models of the underpinning theory and other methods of processing the same language behaviour;
4. to indicate the performance of a model of the underpinning theory when applied to more complex language interaction.

1.5 Research constraints.

The goals associated with this research can only be realised to an extent commensurate with constraints that are either self imposed or determined by other research, for example:

1. to the best of the author's knowledge, there has been no published research that describes the processing of language using fuzzy pattern matching of concrete examples of textual representations of language. Though research methods borrow from other work, the strategy employed here is hand-crafted, novel and not supported by accepted practice for this particular application;
2. computer implementations of the model are at a prototype stage, and no claim is made for elegance of system design or programming. It has been necessary to render the implementations of the underlying theory 'transparent', so that interim machine states during individual discrete processes may be observed. Consequently, the design of implementations is more complex and slower than elegant designs that would be expected of operational models. For example implementations:

    - are modelled at a low level of abstraction to keep system understanding (and thus maintenance and modification) as simple as possible;
    - use slow, visual objects so that interim data can be more easily observed and recorded;
    - use a programming environment (Visual Basic version 3), chosen for its excellent prototyping qualities and visual tools, rather than its speed or power.

The method being investigated in this research is at an early stage of development and there is no implication that it is yet intended to equal the performance of more established approaches.

1.6 Plan of action and document format.

The following plan was compiled during the early stages of research and there has been little deviation from it. The plan is offered in an uncomplicated chronological list. The structure of this thesis corresponds closely with the plan, and the reader is referred to the relevant chapters for elucidation on points of detail.

- research into NLP;
- theorise and design a model of the underpinning theory that can be externalised using a computer programme;
- research into formalisms capable of solving problems within nonmonotonic spaces;

- research into related work;
- hypothesise a research strategy;
- design, implement and test a simple low-fuzzy computer implementation of the model;
- perform and analyse the results of experiments that test the ability of this implementation to process simple language interaction;
- design implement and test two progressively more fuzzy implementations of the model;
- perform and analyse results of experiments that test:
  - the effect of increased fuzzy processing on implementations performance;
  - the relative performance of implementations against some traditional methods of processing the same data;
- analyse the implementations for intrinsic limitations in language processing performance;
- theorise and design a model of the underpinning theory that can be externalised using a computer programme, that may be capable of processing language interaction outside the limitations of previous implementations;
- perform experiments that test for enhanced language performance;
- draw conclusions, and compare research performance with original goals.

1.7 Summary.

Following a preamble, this chapter introduced the context of the research upon which this thesis is based. This section also included an overall hypothesis and a working hypothesis that the research sets out to test. This was followed by a brief description of the major components of the research. An account of the concepts that motivated the work was then given. This was followed by a set of goals that were designed to enable an indication of overall research performance. Some intrinsic limitations upon the implementation that are employed to test the underpinning theory were listed. Penultimately, a research plan was provided that corresponds with the format of this thesis. The chapter concluded with a summary.

2.0 Domain, grounding and motivation.

This chapter begins with an overview of the domain within which Baby research was conducted. The overview includes a high level description of the methodologies available, along with theoretical factors that have limited the performance of applications to date. It is noted that Baby research may allow the possibility of overcoming some aspects of the limitations associated with machines that acquire problem solving behaviour during exposure to large and complex problem spaces. This is followed by an overview of machine learning strategies used in artificially intelligent research, and a positioning of Baby within these strategies. The remainder of the chapter is divided into two parts, each dealing with two related, important and distinct research disciplines. Part one deals with issues relating to NLP, and part two deals with issues relating to reasoning with uncertainty.

Part one, begins with the reasons for choosing NLP as a test domain. An introduction to NLP is then given and problems associated with wide domain processing are highlighted. An overview of previous NLP research is given and this is followed by example applications. The predominance of this overview is devoted to language acquisition machines. Baby research is then explained and a detailed account is provided of how it significantly extends antecedent research. The Baby model is then superficially described. There follows an exposition of related work by other researchers, and an argument is given for how this research significantly differs from that work.

Part two begins with a review of current formalisms employed in designing machines that reason in incomplete problem domains. Formalisms are reviewed with respect to Baby research and an argument is provided for the reasons for choosing possibility theory (manifest in fuzzy logic) for this research. Fuzzy logic is then briefly described. The chapter concludes with a summary of both parts one and two.

2.1 Overview of research domain and current limitations.

Theorising machines, capable of solving problems in domains that are sufficiently large and/or complex as to be not yet fully specified, is a difficult issue facing AI research. This is especially true where problems exist within spaces that are either chaotic and/or dynamic and thus nonmonotonic. Most machines that can navigate these domains have employed one (or a combination) of two strategies for possessing knowledge about their problem space:

- those programmed with knowledge;
- those that are programmed to acquire knowledge.

So far, the predominance of research has concentrated upon employing the former strategy, (see *2.5 Brief background of NLP research.* for some example applications that use each strategy). More specifically, the methods used to implement each strategy are respectively:

1. to program machines with two sets of rules, one that specifies aspects of the problem domain and another that specifies how to apply these rules to solve problems within the domain. A machine so programmed has the potential to hypothesise solutions to novel problems within its domain, by applying the rules to navigate solution spaces.

2. to program machines to store problem/solution behaviour during interaction with the problem domain (usually called training). Such machines have the potential to hypothesise solutions to novel problems using various methods (see section *2.2 Overview of artificial intelligence learning strategies.*).

The first method is manifest in symbolic variants and the second, in both symbolic and subsymbolic variants. For complex domains, all three methods suffer from significant limitations. Illustrations in the following text come from NLP research. This is because the area contains example complex domains and it is relevant to the test domain used in this research.

1. **Symbol systems that are programmed with domain knowledge.**

   Compiling sets of specification and application rules that describe complex problem domains is notoriously difficult. For example, Collier (1994) wrote "... a large proportion of the NLP community do not believe that general principle rules will provide a solution to the language problem".

2. **Symbol systems that are programmed to acquire domain behaviour during interaction.**

   Designers of machines that are programmed to acquire solution behaviour during interaction with complex problem domains, are faced with at least three major difficulties:

   - many behaviours are exceptional, (e.g. Collier, 1994);
   - the predominance of problem/solution sets are positive (e.g. Crain, 1991 or Meier, 1991). It is notoriously difficult to program a symbol system that can acquire domain knowledge from predominantly positive example. Indeed, Gold (1967) showed that a set of rules that specify any non-trivial language, could not be learned from positive examples alone;
   - incompleteness, imprecision and ambiguity are ubiquitous, (e.g. Manaris & Slator, 1996 or Allen, 1995).

3. **Subsymbolic systems (connectionist) that acquire domain behaviour during interaction.**
   Human ability to navigate complex domains suggests that subsymbolic, connectionist approaches may eventually provide a solution. However, machines implementing this methodology, that are capable of processing these domains, are currently impractical on both theoretical and technical grounds. Research into this methodology is "still concentrating on seeking ways of analysing representations as a means of understanding the problems facing automated natural language processing" (Reilly & Sharkey, 1992).

Hybrid systems have also been developed that employ either different configurations of connectionist networks, or the combination of connectionist networks and symbolic systems. This is a *horses for courses* strategy where symbolic and connectionist implementations are mixed and matched, each contributing in areas where their computational power is best suited, (see Wermter (1995) for a good overview). Though the approach shows promise, e.g. RINA (Zernik, 1987) or CHILD (Selfridge, 1980), it is argued here that ultimately they may suffer from combinations of the problems facing individual methodologies shown above.

It was an awareness of these limitations and a hypothesis that this research may circumvent some of those that relate to machines that acquire domain behaviour during interaction, that provided the major motivation for pursuing the work, (see *2.6 The context of this research within the NLP discipline.* for more details).

2.2 Overview of artificial intelligence learning strategies.

Baby is presented as a novel approach to machine learning within a domain normally processed by intelligences. Natural language machine learning is discussed in more detail later in this chapter, but some basic principles are introduced here. According to Michalski (1992), there are a number of different strategies that have been employed to model intelligent machine learning. The strategies are précised in the following list.

1. **Rote learning:** system responds with solution to identical, previously experienced problem.
2. **Learning by instruction:** small amounts of processing are involved, including the representation of knowledge as internal symbols.
3. **Learning by deduction:** truth preserving inferences are derived from learned knowledge thereby generating meta-knowledge that is used to deduce solutions.
4. **Learning by analogy:** novel problems are classified as similar to previous problems and knowledge associated with these previous problems is used to process novel solutions.

5. **Inductive learning**: similar to deductive learning except that the meta-knowledge generated is inferred from weaker truth derivations and is thus more powerful (if less reliable).
6. **Learning from example**: the induction of concepts from positive example causing the concepts to become more general, and in some cases, the use of negative examples causing the concept to become more specific.
7. **Learning by observation**: identifying general principles and similarities from observation. The method is totally unsupervised and relies on inferential processes.

These strategies are ordered in an ascending list manifesting a shift of learning emphasis from tutor to machine. Baby can be thought of as being a member of the set 'Learning from example' (*6* above) and exhibit some aspects of the set 'Learning by observation' (*7* above).

The remainder of this chapter is divided into two parts. The first part deals with the test domain used in Baby research. The second part deals with an important and distinct issue that emanates from part one, that of reasoning with uncertainty.

Part One: The test domain: natural language processing.

Part one begins with the reasons for choosing NLP as the test domain for this research. The NLP discipline is then introduced and reviewed from the perspective of language domain size vs. processing method. A brief review of NLP applications is then given, both from the perspective of machines that are programmed with domain knowledge and then from those that acquire domain knowledge. There follows a description of the context of this research and how its antecedent research has been extended. This is followed by a superficial description of the Baby model. An account is then given of related research and how Baby research may be distinguished from it.

2.3 Choice of test domain.

In order to design, implement, test and measure the performance of externalisations of models based upon the general theory of AI posited in the previous chapter, it was necessary to chose a domain of intelligence to which it could be applied. Literature that purports to provide overviews of the AI discipline, contains lists of such domains. Earlier works such as Boden (1977), for example, identify language understanding, vision, learning, creativity and problem solving. Later works such as Sharples, Hogg, Hutchinson, Torrance & Young (1990) add knowledge elicitation, search and reasoning, whereas more recent works such as Rich & Knight (1991) include game playing, planning, and common sense. The trend in current works seems to be application based. This is

presumably a result of successful research work in earlier more general areas, e.g. Jang, Sun & Mizutani (1997).

For an activity requiring human intelligence to be suitable for the underpinning theory to model, it would be preferable if it possessed some specific attributes:

1. intelligent, easily identifiable and self-contained behaviour possessed by most people;
2. overt, easily identifiable and detectable patterning of afferent and efferent stimuli;
3. simple to discriminate between intelligent and dumb behaviour;
4. beneficial to many people if computers could simulate the intelligence.

The rationale in each case was to:

1. maximise the scope for a wide range of application of the underpinning theory;
2. be relatively simple to implement;
3. be relatively easy to measure performance;
4. maximise the possibility that application of the theory will have social value.

All the AI domains mentioned above, plus others investigated such as robotics, expert systems, speech recognition/production systems and risk management systems, comply more or less with attributes 1 & 4. Attribute 2 is more difficult because the patterning of afferent and efferent stimuli is often difficult to detect, e.g. planning or game playing. Attribute 3 is even more difficult to assess. Can one reliably discriminate between intelligent and dumb creativity for example? The list of candidate domains was eventually narrowed to two; vision or language. However, some work on the lines described here has already been done in the field of vision, e.g. Anderson (1990); Biederman (1987) or Marr (1982).

Consequently, human language processing appeared to be a suitable candidate for the underpinning theory to model. This is because natural language:

1. is an intelligent behaviour (axiomatic);
2. is used by all people and is easily distinguished from other activity; though some argue that language is not a separate activity but part of a continuum that characterises human interaction in general. In NLP this is manifest in multi-modal approaches (see McKevitt (1996) for an overview);
3. exhibits easily detectable, patterning especially in the written form (axiomatic);
4. is easily distinguishable from random burble by another user of the language (axiomatic);

5. if processed by computer with anything like the competence that humans possess, would benefit many people in many different and diverse areas. It is also argued however, that people do not wish to talk with computers as they talk with each other, but prefer to use heavily constrained sub languages e.g. Diaper (1986) or Diaper & Shelton (1987).

In addition, NLP has proved a very difficult AI problem to solve, and this challenge supplied considerable motivation to its choice.

2.4 An introduction to natural language processing.

A comprehensive account of NLP would be lengthy and outside the scope of this thesis. Instead, an overview is given here that considers some of the issues involved particularly from the perspective of its application in the Baby model, i.e.:

- a description of the natural language domain;
- implications concerning the machine processing of wide sub-domains;
- brief background of NLP research with examples of machines that are programmed with domain knowledge;
- theory behind and examples of machines that are programmed to acquire domain knowledge.

2.4.1 Domain description.

The term *natural language* subsumes all human language. Domains of natural language are called sub-languages. Natural language supports a domain far larger than any of the applications so far purporting to process it. In the author's opinion, the term NLP is, therefore, a misnomer for the discipline with which it is generally associated. The title is better thought of as an expression of the discipline's goal rather than a description of its activity and product. However, the discipline has been so named for many years and the term will be adopted here for the sake of conformity. According to Sparck-Jones & Galliers (1995), at least five sub-languages can be hypothesised to describe various levels of language complexity. Table 1 shows an adaptation of her analysis, with texts that are typical of sub-language style:

natural language	:	what this thesis is written in
full language	:	weekly magazines
sublanguage	:	medical reports
serious material	:	scientific articles
trivial material	:	food label ingredient list

Table 1: Some sub-languages with example styles (adapted from Sparck-Jones & Galliers (1995))

Due to the size of its domain, NLP can only be viewed from a particular perspective in limited textual descriptions such as this. It will be considered in this section from the perspective of how the range of language, and the knowledge required to use that range, affect the modelling techniques employed to process it. The rationale being, that the approach described here, is hypothesised to possess wide domain potential, (see sections *2.6 The context of this research within the NLP discipline.* and chapter *9.0 Summary, general discussion and conclusions.* for more details)

2.4.2 Domain size vs. Processing methods.

Programming computers to process minor behavioural aspects of language can be a trivial exercise. To use an extreme example to illustrate the point, one line of code is required to simulate the behaviour of always answering "Yes." to any typed question (as denoted by a String ending with the character "?"). When expected to process larger domains of language behaviour, the exercise suffers combinatorial expansion.

For wide domain applications both symbolic and connectionist (and consequently hybrid) implementations suffer from significant limitations (see *2.1 Overview of research domain and current limitations.*)

The community's strategy for overcoming the problems associated with either methodology, is to apply the methods to narrow domains of both language and knowledge. Earlier applications that illustrate this strategy include Winograd's (1972) SHRDLU, where language and knowledge domains are restricted to a bounded universe of coloured blocks. Or more recently, The 'Lo' project (Feldman, Lakoff, Bailey, Narayanan, Regier & Stockle 1996), that employs domain restriction to a specific image. Some systems disguise the existence of their restricted language and knowledge domains by processing out of domain input with elicitous 'fall-back' replies, such as the "Tell me more about .....?" reply structure used by Weizenbaum (1966) in his Eliza program.

However, "*the benefits gained by possessing machines having the ability to perform*...[wide domain]...*natural language processing cannot be over-estimated.*", (Joshi 1991 and again in 1993). Since these publications, the language environment has been extended due to the exponential increase in electronic textual information. Consider our enhanced information processing power, if machines were able to assimilate knowledge by reading large volumes of text, précising, prioritising and presenting it from the perspective of particular contexts or different languages.

A new and interesting avenue that addresses the issue of wide domain language processing is that of corpus-based methods, see Ney, Steinbiss, Haeb-Umbach, Tran & Essen (1994) or Ney, Essen & Kneser (1994a) for a good account. It will become apparent that the work described here can be associated with this approach in the guise of Data Oriented Language Processing (see section *2.8 Related research.*). However, generally speaking, "*natural language is complex and diverse and machines which model its use remain difficult to design*" (Collier, 1994). NLP applications must deal with complexities such as:

- **inaccuracy**, e.g. errors in spelling or pronunciation, transposed words and ungrammatical utterances;
- **incompleteness**, including elliptical constructs like *"Me too."* in response to *"I know the secret of the universe."* and anaphora like *"They are green."* in response to *"What colour are the traffic lights?"*;
- **imprecision**, including the use of terms without a specific reference point such as *"tall"* in *"George is very tall."* and qualitative terms such as *"beautiful"* in *"Mary is beautiful."*;
- **ambiguity**, in which multiple interpretations may arise at any level of linguistic knowledge, (examples of ambiguity are developed from Manaris & Slator, 1996; Obermeier, 1989 and Cottrell, 1989).
  - **acoustic**, as in *"It's hard to wreck a nice beach."* (recognise speech);
  - **lexical**, as in *"The pen is large."* (pig house or writing implement);
  - **phrasal**, as in *"Old men and women."* (old men, old women, both old?);
  - **sentential**, as in *"I saw the man on the hill with the telescope."* (who has, and what are they doing [if anything] with the telescope?).

(developed from Allen, 1995)

A diagrammatic representation of the continuum of inter-relationship between domain size, language type, computational technique and complexity is shown in Figure 1.

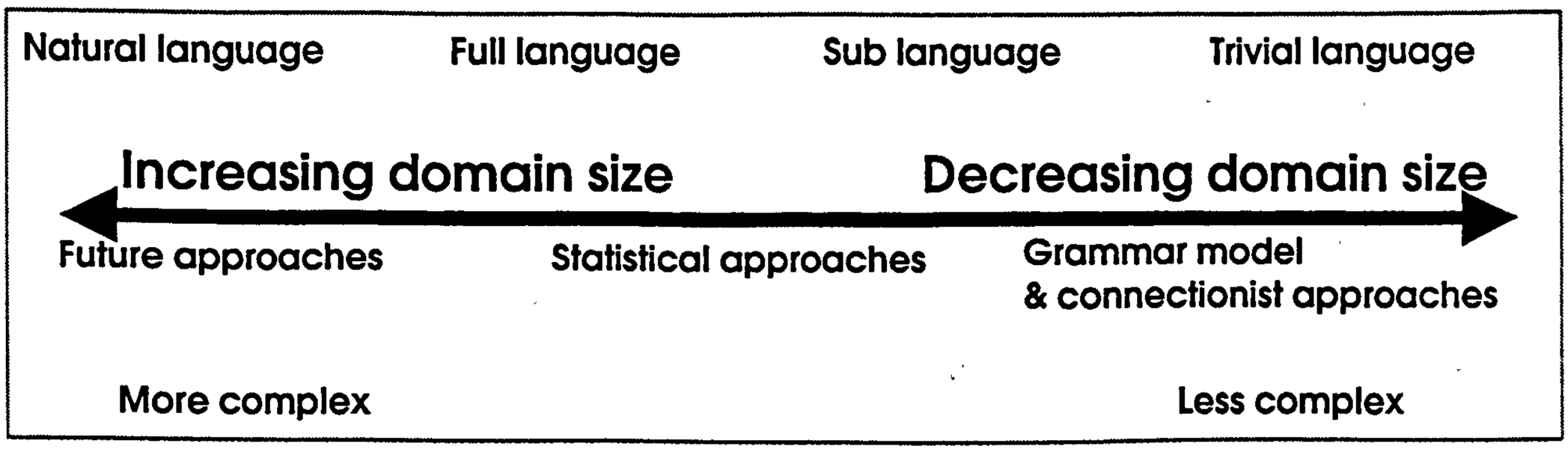


Figure 1: Relationship between domain size, language type, computational technique and complexity

2.5 Brief background of NLP research.

Processing natural language by computer originated some 50 years ago. During this period, the discipline can be viewed as having evolved through three eras; pre-theoretic, theory-laden and user-centred (Manaris & Slator 1996). Pretheoretic approaches applied simple dictionary based look-up solutions to the processing of natural language. The approach failed, because the theoretical basis for such simple modelling inadequately explained the richness of linguistic complexity.

The reasons for the failure of the pre-theoretic era motivated the theory-laden era. Here, more powerful and explanative models of language were researched. The era saw an increase in experimental work conducted upon an ever growing number of grammatical, logical and semantic theories. The research produced a plethora of *toy* systems that were implemented to illustrate the validity of various narrow aspects of language and meaning, e.g. Anderson (1977), Granger & Foulu (1977), Kolodner (1980), Langley (1982), Langley & Neches (1981), Selfridge (1980 & 1981), Siklossy (1972), Siskind (1990) and Zernik (1987). Over the past ten years, emphasis has turned to a user-centred era. Here, the narrow domain problem is accepted and applied to fit particular market niches. Example applications that have been researched during this era include:

- spelling checkers;
- grammars and style checkers;
- portable multilingual dictionaries;
- internationalisable software;
- machine translation;
- natural language database systems;
- text interpretation (including):

- information retrieval;
- text categorisation;
- extraction of data from text.

These practical applications share at least two properties. Firstly, "*.... they focus on a particular domain*", and secondly, on "*a particular task, rather than attempting to understand language completely.*" (Russel & Norvig, 1995).

2.5.1 NLP systems that are programmed with domain knowledge.

There follows a résumé of some more notable applications that have been researched, successfully implemented and employed within three major areas of NLP that are programmed with domain knowledge.

2.5.1.1 Machine translation.

Machine translation (MT) was one of the first major applications of computer based NLP. Locke & Booth (1955) give a good account of those early, enthusiastic days. Researchers soon became disillusioned with their attempts to implement practical MT systems. The reasons for their failure is well described in Bar-Hillel (1960). The consequential need to model world knowledge in MT implementations is introduced in Lindsay (1963). Taum-Meteo, developed at the University of Montreal, could translate weather reports from English to French. It is one of the more famous practical MT systems that was spawned during post 1963 research and is well documented in Quinlan and O'Brian (1992). Another is SPANAM (Vasconcellos & Leon, 1985), that could translate more generally from Spanish to English, albeit in rather an ungrammatical and non-fluent manner. For more recent MT applications, the reader is referred to Voorhees (1993) who describes implementations based upon Wordnet.

2.5.1.2 Natural language database systems.

Possibly the most famous of natural language database search systems is the LUNAR project (Woods, 1972), built for the NASA Manned Spacecraft Centre. LUNAR enabled geologists to ask questions in restricted natural language of a database that stored data concerning lunar rock and soil samples recovered during the Apollo missions. CHAT (Pereira, 1983) is another oft-quoted application that accepts language queries on a database containing world atlas information. One major problem with these applications, and most of their contemporaries, is their inability to span meaning over the domain of individual queries. TEAM (Grosz, Appelt, Martin & Pereira 1987) was

one of the first applications to address this issue. Later applications of natural language database systems have tended to utilise statistical methods; the reader is referred to Young & Bloothooft (1997) for a comprehensive review.

2.5.1.3 Text interpretation.

Text interpretation comprises three sub-domains;

- information retrieval;
- text categorisation;
- data extraction.

The task of information retrieval is to choose from a set of documents, the one(s) that are relevant to a query. Text categorisation, on the other hand, sorts text into topic categories, whilst data extraction takes (usually) on-line text and derives from it assertions that can be loaded into a database. The SCISOR system (Jacob & Rau 1990), that analysed news stories, is a prime example.

2.5.2 NLP systems that acquire domain knowledge through interaction.

Due to the significance of machine learning to this research, this section deals with language acquisition systems in more detail than with systems that are programmed with domain knowledge. See Collier (1994) and Gorin (1995) for two good papers on this subject.

For machines to learn human language through interaction they must acquire at least two kinds of knowledge. Firstly, a knowledge of the structure (grammar) of the language and secondly, what to say and when to say it (see competence vs. performance distinctions drawn in Chomsky (1965)). Research into language learning systems deal with either the former or both tasks. Fundamentally, there are four basic methodologies that have been successful in some degree (adapted from Collier 1994).

1. **Syntactic approaches:**

   These systems acquire the grammar of a language from exposure to positive example domains containing grammatical sentences. Usually, such grammar rule sets classify words by function (syntactic categorisation) and sequence patterns of word function (re-write rules) to describe legal sentences and sub-sentence (phrase) structure.

2. **Semantic approaches:**

These systems acquire conceptual representations of sentence meaning. One advantage of such systems is that they are able to categorise sentences with different structures but similar meaning. Example semantic representations are Preference Semantics (Wilks, 1973) or Conceptual Dependency (Schank, 1973).

3. **Pragmatic approaches:**

These systems are similar in principle to semantic approaches, except that conceptual representations are grouped into hierarchies. Members of such hierarchies contain meta-knowledge inasmuch as, individual concepts also acquire inherited knowledge from connected nodes. Examples of such conceptual hierarchies are frames (e.g. Minsky, 1975) and semantic networks. Temporal, hierarchical ordering is another common grouping of conceptual representations. Such entities contain the meta-knowledge of the sequence (or chain) of events to be represented. An example of such groupings is scripts (e.g. Schank and Abelson, 1977). One major advantage of both frames and scripts is that they enable disambiguation as a consequence of individual concepts existing within a contextual framework.

4. **Connectionist approaches:**

Connectionist approaches are fundamentally different from other approaches, inasmuch as knowledge is subsymbolically represented across networks of weight connected nodes. Each training epoch comprises exposing the network to a training set comprising positive example stimuli. Following each stimulus, the network hypothesises a response, the accuracy of the hypothesis is evaluated by comparing it with the correct response that is also contained within the training set. Each connection weight is then mathematically adjusted in order to reduce error (e.g. back propagation [e.g. Rumelhart, Hinton & Williams, 1986]).

2.5.2.1 Example syntactic language acquisition applications.

These type of machines may generally be classified into those that learn the structure of formal languages and those that employ models of aspects of cognition to acquire human language behaviour. Cognitive implementations are mostly concerned with modelling aspects of child language acquisition. Earlier machines often adopted the formal approach, see Pinker, (1979) for a good overview, whereas later machines often adopted the cognitive model approach.

Some examples of cognitive model approaches include, Kelley (1967), who utilised a cognitive model of child syntactic acquisition. Hill (1983) researched a system that purported to acquire language to the level of a two year old child, and MacWhinney (1987) produced a system that, when

seeded with a single word, learned to build more complex linguistic structures. Other notable language acquisition research include Problem-Solving Theory (PST), (Reeker, 1976), the Language Acquisition System (LAS), (Anderson, 1977) and AMBER (Langley, 1982).

2.5.2.2 Example semantic/pragmatic language acquisition applications.

Fewer systems have been researched that utilise this approach than by the syntactic approach described above. Early models include ZBIE (Siklossy, 1972), that acquired Russian language constructs from propositional representations of their semantics. Also, Hedrick's Production System (Hedrick, 1976), converted sentences into semantic network representations. Other semantic/pragmatic systems of note are FOUL-UP (Granger and Foulu, 1977), MORAN (Salveter, 1979), CYRUS (Kolodner, 1980), CHILD (Selfridge, 1980), GENISIS (Mooney, 1985), RINA (Zernik, 1987), MAIMRA (Siskind 1990), EBNLA (Liu, and Soo 1994) and the L(0) project (Feldman, Lakoff, Bailey, Narayanan, Regier and Stockle, 1996)

2.5.2.3 Example connectionist learning language acquisition applications.

Artificial neural networks (ANNs) were first legitimised by McCulloch and Pitts (1943), however natural language acquisition using such mechanisms is comparatively recent. This was due to the community's ignorance of training algorithms suitable for, multi-layer networks (e.g. the perceptron convergence theorem (RosenBlatt, 1958) vs. back propagation methods (Rumelhart, Hinton and Williams, 1986b). Consequently, early ANN research concentrated upon simple, single layer networks, and this architecture was capable of representing the domain of only very simple language processing. Serious connectionist NLP began in the mid 1980s with Rumelhart and McClelland (1986), who developed a multi-layer connectionist system that, they claimed, accounted for the three stage acquisition of the past tense of verbs observable with children, (e.g. Brown, 1973, Ervin, 1964, Kuczaj, 1977). Concerns regarding the experimental conditions of Rumelhart and McClelland (1986), highlighted in Rumelhart, Hinton and Williams (1986a) spawned another famous connectionist language acquisition model PARSNIP (Hanson and Kegl, 1987). Recursive Auto-Associative Memory (RAAM) (Pollack, 1990) was a connectionist language acquisition machine that attempted to address a problem generally associated with connectionist models, that of analysing and understanding the distributed representation created during the learning process. Another interesting connectionist research that used language acquisition as a test domain was Elman (1990); this research utilised recursive networks. The main difference between this work and other previous research was the method by which Elman's networks were configured to encode the temporal dimension. Elman achieved this by arranging his networks such that new stimuli in a temporal sequence of stimuli were mixed with representations created by previous stimuli in the sequence.

Many other methods up until this time had used sliding window approaches to encode time (e.g. NETTALK (Sejnowski and Rosenburg, 1987) or PARSNIP (Hanson and Kegl, 1987). One reason for describing Elman's work is that Baby exhibits similar behaviour to aspects of his networks, albeit by a different route, (see *7.4 Results of processing text with no white-space.* for more details)

2.6 The context of this research within the NLP discipline.

Despite the amount of research aimed at producing machines that can process and/or acquire language (of which section *2.5 Brief background of NLP research.* gives only a selection), the basic problems (outlined in *2.1 Overview of research domain and current limitations.*) have so far prevented the community developing a machine that can process language as adequately as most humans. Consequent to this state of affairs, a significant motivation behind pursuing Baby research was the principles underlying some unpublished, part-time research entitled 'Sentence Encoder And Reply Selector' (SEARS), whose underlying theory hypothesised the possibility of overcoming some of the limitations suffered by other methods of processing wide domain human language, (Lefley, 1993).

2.6.1 SEARS.

At the outset of this research, the concepts behind SEARS were not as concrete as this subsection suggests (see relatively recent citation dates). In retrospect however, their formalisation added little to the substance of the SEARS concept. SEARS may be is summarised as follows, (précised and adapted from Lefley and Rogers (1997) and Rogers and Lefley (1997)).

Acknowledging that humans are good at solving complex problems, SEARS is motivated by the notion that cognition (particularly at low levels) can be thought of as a pattern association process, e.g. Anderson (1990), Eysenck (1984), Ross (1995), Smyth, Collins, Morris & Levy (1994). SEARS detects and stores concrete patterns of behaviour from problem/solution sets of a domain to which it is exposed and uses these patterns to compute hypothetical solutions to novel problems from the domain. SEARS is not programmed with rules that model the problem domain itself. Instead, a more flexible modelling, employing context sensitive pattern rules is used. This may allow, SEARS to use the problem domain, to supply solution strategies, generated directly from the complexity of the domain itself.

From the outset of this research, Dr. Lefley and the author agreed that SEARS utilised the connectionist paradigm. For example, the SEARS software:

- used raw data throughout all processing stages;
- detected and stored common patterns in such data;
- associated the pattern of new input with its database of patterns;
- produced output by processing patterns derived from new input with stored patterns;
- underwent learning strategies;
- hypothesised contextually appropriate solutions to novel problems.

However, in contradistinction with research normally subsumed by this methodology, in SEARS research, it was also agreed that neural processes were modelled at a higher level of abstraction than the biological, yet not so abstract as to embody the instantiation of rules that model a theory of the nature of language. In other words, SEARS simulates aspects of the ***behaviour*** rather than the ***mechanism*** of neural networks. We theorised that this approach may capitalise on the strengths of connectionist strategies, without invoking some of the problems of constructing and teaching hugely complex ANNs, or relying on rule-based specification of high level theories of language behaviour

2.6.2 Extending SEARS to Baby.

According to Dr. Lefley, the original, ultimate aim for SEARS was:

> *to develop a language interpretation system that derives a valid, flexible grammar by exposure to that language in a similar way to that of a human child. The hypothesis is that a fuzzy learning system can adapt to the nuances of human language in better ways than other more rigid automated solutions. The learning system is designed to be flexible enough to learn other domain specific grammars such as foreign languages, scientific or computer dialogues. Learning systems can also adapt to a specific user, application or situation.*
>
> (published latterly in Lefley and Rogers, 1998a)

In 1995, Dr. Lefley invited the author to progress this research as part of a PhD project. At that time, the majority of SEARS research had comprised theoretical consideration of an alternative NLP method. Much of Dr. Lefley's motivation came from the notion that it may be possible to utilise pattern matching strategies already used with some success in hypothesising image identification (e.g. Anderson, 1990; Biederman, 1987 or Marr, 1982) to hypothesising responses from textual representations of human language stimuli. In addition to these theoretical issues, Dr. Lefley had also written a piece of software that modelled some aspects of the original aim.

Dr. Lefley's work specifically did not include:

- a formal underlying theory;
- in-depth, rigorous modelling;
- software performance tests;
- modelling of the fuzzy aspects of the original aims;
- language performance metrics.

The research described in this thesis can be considered to be an investigation into theorising, designing, constructing and evaluating a machine that extends the work already done with SEARS on the areas itemised above. More specifically, the extension comprised:

- formalising the underlying theory;
- producing an in-depth and rigorous theoretical model of the underlying theory;
- instantiating a nonfuzzy application of the model on a computer;
- ensuring adequate software performance;
- conducting language performance test with the application;
- incorporating fuzzy modelling into the application;
- conducting fuzzy vs. nonfuzzy language performance metrics.

The result of work concerned with item 1 above was manifest in the underlying theory stated in *1.1 Context of research.*. There follows a description of work concerned with item 2 above. All other items are addressed in other chapters of this thesis.

2.7 The Baby model.

This section begins with a brief discussion on some of the philosophical plausibility issues relating to the research described in this thesis. This is followed by an explanation of the components used to model and externalise the underpinning theory. Assumptions made and heuristics employed are then stated. There follows a description of how the model components are synthesised to form a high level system specification. Finally, a summary is given of the more important points raised, along with some meta-phenomena that emerge in complex application of the model.

2.7.1 Philosophical plausibility.

An exposition on the philosophy of AI in general, is outside the scope of this thesis. Instead, the issue of philosophical plausibility of the genre of machines of which the Baby concept is an instance is briefly considered.

A mechanism that externalises the Baby concept is subsumed by the set of AI machines. Many commentators assume that in any normal sense, intelligent action is the consequent manifestation of a thinking process. This concept was first posited by Aristotle in what he called "right thinking"; an irrefutable reasoning process comprising the specification of patterns of argument structures that always give correct conclusions assuming correct premises. If thinking can be proved to be solely a human attribute, then the aims of this project cannot be realised. Despite extensive debates on the matter, it is argued here that no such proof has been presented. The attitude adopted for this research mirrors that of the philosophers in Dennett (1984), where he rather more metaphorically, whimsically and strongly than is presented here, states:

> *It is rather as if philosophers were to proclaim themselves expert explainers of the methods of a stage magician, and then, when we ask them to explain how the magician does the sawing-the-lady-in-half-trick, they explain that it is really quite obvious: the magician doesn't really saw her in half; he simply makes it appear that he does. "But how does he do that?" we ask. "Not our department," say the philosophers.*

AI scientists, cognitive scientists and philosophers are trying to resolve many of the same questions such as: how *can* minds work?, how *do* human minds work? and can non-humans have minds? AI and philosophy have often assumed adversarial roles, but recently there seems to have been a trend by some philosophers to "*side with the computational approach provided by artificial intelligence, partly because it has the tools and the inclination to give detailed, causal explanations of intelligent behaviour*" (Russel & Norvig 1995). In part, due to this softening of philosophical attitude and, in part, due to personal conviction, the stance adopted for this research coincides in all respects, except time scale, with the position taken by Turing (1950) where he asserts:

> *I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.*

2.7.2 Modelling components.

This subsection give an explanation of the components used to model and externalise the underpinning theory. Following research into cognitive psychology in general, and given the theory that underpins the Baby concept, it was decided to externalise the concept by modelling some aspects of five attributes manifest in human neural processing, some of which are also observable in ANNs and many symbolic models of human behaviour:

- attention;
- forming hierarchical structures of patterns of data;
- pattern recognition;
- activation;
- reasoning with uncertainty.

By modelling aspects of human cognition, it is not intended to imply a strong AI approach. No argument will be presented suggesting that an externalisation of the Baby concept may result in an 'intelligence', or necessarily bear any relation to mechanisms employed in the brain. Rather, it is hypothesised that an externalisation of the Baby concept may result in a machine that acts as if it possessed human linguistic intelligence. Aspects of the five behaviours modelled, are discussed in following sub-sections.

2.7.2.1 Attention.

Attention is a complex human behaviour, but an in-depth exposition lies outside the scope of this thesis, see Anderson (1990) for a good account. The aspect of attention modelled by Baby, is that it attends to the largest subset of stimuli within a supra set of competing stimuli. This is an important concept that pervades Baby models. The concept will hereafter be referred to as the *'concentrate on the largest principle'* (*COTLPin*).

2.7.2.2 Patterns of data.

Processing patterns in data is arguably one of the most pervasive characteristics of both human cognition and AI machine processing. In humans, the notion that patterns of sensory data play a role in cognition occur regularly in 80s literature, e.g. Dreyfus & Dreyfus (1986), Edleman, (1985), Holland, Holyoak, Nisbett & Thagard (1986), Lakoff (1986) or Rumelhart and McClelland (1986).

A researcher who posited the notion that pattern processing constitutes a central role in cognition was Howard Margolis, who argued a view that "*everything is reduced to pattern matching*"

(Margolis 1987). Extreme though this stance may seem, opinion has moved toward centralising the role of patterns rather than superseding them with other entities, e.g. Anderson (1990), Ross (1995) or Smyth et al. (1987). Most models of cognition utilise pattern processing in the form of extraction and/or recognition as a part of their process. In symbolic implementations, pattern processing manifests itself in artefacts such as slot-and-filler structures, see Rich and Knight (1991) for a good overview. In connectionist implementations, though its manifestation is not explicit, pattern association is implicitly involved, even characterising ANN behaviour, such as with pattern associators, e.g. Ling, (1994). Baby theory models two aspects of pattern processing as follows:

2.7.2.2.1 Forming hierarchical structures of patterns of data.

Hierarchical pattern structures appear overtly in symbolic and covertly in connectionist models. Examples of symbolic hierarchical pattern structures include scripts, frames and semantic networks. In frames for example, hierarchical layers are structured by *isa* and *instance* relationships. In the case of connectionist models, in the author's opinion, layers within a network can be thought of as hierarchical levels of connectionist representation of the training domain.

2.7.2.2.2 Pattern recognition.

A significant proportion of the information to which people are exposed, is presented in the form of patterns of data. Beale and Jackson (1992), for instance, expound the pervasive nature of pattern recognition in both human and ANNs, illustrating their point with the example of reading, saying:

> *The text that you are reading now is presenting you with complex and varied patterns in the form of strings of letters. Before we even start to consider the far reaching cognitive issues of language processing, the visual system must first solve the pattern recognition problem. That is, recognising the pattern of letters in the neatly aligned ink stains on the page.*

Pattern recognition can be defined as a process of identifying structure in data by comparing it with previously experienced structure. Generally, pattern recognition can be considered as a two stage process:

- feature extraction;
- classification.

A feature can be defined as some metric of the input pattern. The metric is chosen such that it characterises the input type. Once the feature extraction process has compiled a list of feature measures, the classifier attempts to map an input to one (some) of the features and in so doing hypothesises the classification state of the input.

2.7.2.3 Activation.

Activation is a process whereby individual nodes in a network compute their states according to activation from the output of their daughter nodes. The state of so-activated nodes, determines whether they, in turn, provide activation to their parent nodes. Activation is common to both strong and weak implementations of connectionist models and is thought to play an important role in biological neuron computation, e.g. Anderson, (1990) or Fischbach, (1993). Examples in strong connectionist implementations include perceptron networks (RosenBlatt, 1962) and Hopfield networks (Hopfield, 1982). Examples in weak connectionist implementations include semantic networks, e.g. pandemonium models (Selfridge, 1959 and Selfridge & Neisser, 1960).

2.7.2.4 Reasoning with uncertainty

An important characteristic of an intelligence is its ability to reason with uncertain knowledge, see Krause and Clark (1993) for a good account. The way in which the Baby concept manifests aspects of reasoning with uncertainty is a complex issue and forms a significant element of the research to which this thesis relates. Part two of this chapter covers the issue in some depth. Within this sub-section, discussion will be restricted to why it is necessary to employ reasoning with uncertainty in order to model the underlying theory. Given Baby's underpinning theory, (see section *1.1 Context of research.*), reasoning with uncertainty is required to model the concept represented by the words "most closely correspond". Once an externalisation of the Baby concept has detected and stored textual patterns associated with language experience, it is ready to synthesise response to novel utterance. The method used to detect the patterns of previous experience that most closely correspond to the current stimulus, is not necessarily a deterministic process. This is because it is not possible to compile a general specification for the concept represented by the words "most closely correspond". In addition, for any given non-general specification, the set of patterns that correspond with a current stimulus may not be crisp.

2.7.3 Assumptions.

In addition to an implicit assumption in the validity of the underlying hypothesis, there remains one major assumption associated with this research, i.e. that:

- human language behaviour may be simulated by processing its manifestation in the uni-modal domain of textual representation.

By their very nature, assumptions need not be justified. However, assumptions should exhibit at least a degree of axiomaticity and it is recognised that in accepting this assumption the autonomy of language debate is invoked. On the other hand, the assumption gains credibility when recognising the ability of humans with severely impaired sensual modality to acquire language. This phenomenon may be exemplified with congenitally blind humans, e.g. Anderson, Dunlea & Kekelis (1993), Dimcovic and Tobin (1995), McConachie (1990), McConachie and Moore (1994), or Perezpereira (1994).

2.7.4 Synthesis of model components.

This sub-section describes how the model components: *attention, forming hierarchical structures of patterns of data, pattern recognition, activation* and *reasoning in uncertainty*, are synthesised in an externalisation of Baby's underpinning theory. A simple worked example runs serially with the description. The example utilises natural language representations of the mathematical notation of addition. It employs the general form:

"How is x plus y written?" → "x plus y is written x + y".

There is no linguistic significance in this example of interactional discourse. It was chosen because it satisfactorily illustrates the principles that the Baby concept uses to process language. The following description is not intended to be exhaustive, but to provide an overview of the operation of Baby. A rigorous account of the processes involved is to be found in chapter *4.0 NonfuzzBabe.* Start conditions assume:

- a computer implementation that models the Baby concept that has not as yet been exposed to textual representations of interactional discourse pairs;
- a corpus containing two examples of such textual pairs;

  i.e. **How is 1 plus 2 written?**

  1 plus 2 is written as 1 + 2

  How is 6 plus 7 written?

  6 plus 7 is written as 6 + 7
- a textual representation of an interactional stimulus that is similar to, but disjoint with, the example corpus.

  i.e. **How is 5 plus 9 written?**

2.7.4.1 The pattern extraction process.

Pattern extraction can be thought of as an unsupervised training process. The objective is to detect textual patterns that characterise utterances that form the training set. The number of character patterns shared between two Strings (particularly those with similar structure), is very large. *COTLPin* and *forming hierarchical structures of patterns of data* model components, are utilised to detect a hierarchically structured set of the largest chunks of text that exist between each stimulus utterance and also between each response utterance of the interactional pairs in the training corpus. A set of heuristics can be devised that specifies the application of *COTLPin* and *forming hierarchical structures of patterns of data*, as follows:

1. chunks must be concatenated;
2. larger chunks take precedence over smaller chunks;
3. chunks cannot exist within, or overlap, other chunks;
4. chunk length must be greater than one character;
5. larger chunks prescribe the hierarchical structuring of smaller chunks;
   i.e. chunks partition a String into a hierarchical structure of chunks and non-chunks, e.g. in the part analysis:

Danny[believed it was]windy.
Linda[believed it was]sunny.

further chunk detection is restricted to **'Danny'** with **'Linda'** and **'windy'** with **'sunny'** resulting in a final analysis of:

[Da]nny[believed it was]wind[y.]
Lin[da][believed it was]sunn[y.]

thus precluding analyses such as:

Da[nny][believed it was]w[ind]y.
L[ind]a[believed it was]su[nny].

Chunks between two Strings that comply with this set of heuristics are called constant chunks, or simply constants. When this process is applied to the example training corpus, by comparing all stimuli utterances and separately all response utterances, the following constants can be identified:

][How is][plus][written?][
][plus][is written as][+][

The first string can be considered as the pattern of constants of the stimulus utterances and the second as the pattern of constants in the response utterances in the training corpus. Such strings are referred to as Signatures. Once all of the interactional pairs in the training corpus have undergone this constant identification process, then each Signature is compared with each utterance of the training corpus. If the Signature can be removed from a training utterance, then the textual elements remaining are referred to as variable elements or simply variables. Each different variable is labelled with a different symbol. These symbols are inserted into the placeholder where the variable would have occurred in the utterance from which the pattern could be removed.

i.e. **]α[How is]β[plus]χ[written?]α[**

]β[plus]χ[is written as]β[+]χ[

where: **α = "": β = "one" or "six": χ = "two" or "seven"**

Signatures annotated with symbolised variable slots are called Features. Features derived from conversationally interactive pairs are called FeatureCouplets of the conversationally interactive pairs, or simply FeatureCouplets. FeatureCouplets are stored in a database of FeatureCouplets. Utterances of the training set that formed FeatureCouplets are deleted at the end of the pattern extraction process.

2.7.4.2 Conversationally interactive processing.

Once Baby has performed pattern extraction on a training corpus and processed a set of FeatureCouplets, it is now ready to hypothesise interactional responses to textual representations of novel human interactional stimuli. In view of the small training set used in this example, Baby's language domain is limited. In this simple exposure, conversationally interactive processing is illustrated using the novel textual stimulus **"How is 5 plus 9 written?"**.

Firstly, Baby compares the input utterance with each Signature associated with Features in its database of FeatureCouplets and detects the Feature containing the Signature that most closely resembles the input utterance (if one exists). The metric used to evaluate this resemblance forms a significant part of this research, but for now the simplest of the metrics used will be employed to illustrate Baby processes. This simple metric prescribes that an input utterance resembles a Signature, if all the constants of the Signature appear in sequential order left to right in the input utterance. This metric is called constant equivalence. Membership of the constant equivalence set is crisp. The process of finding Signatures that resemble utterances manifest the *pattern recognition* model component. If the membership function of the resemblance set were fuzzy (that is the case in versions of Baby described in this thesis) then the fuzzy processing required to create the

resemblance set manifests the *reasoning with uncertainty* model component. In this simple example, there is only one FeatureCouplet generated by the training set, and the novel input utterance has been contrived such that it shares constant equivalence with a Signature associated with one of its Features.

i.e. How is 5 plus 9 written? and]α[How is]β[plus]χ[written?]α[

The next step is to remove the Signature that exhibits constant equivalence, from the input utterance. This process establishes variable elements of the input utterance as follows:

i.e.][How is]5[plus]9[written?][gives α = "": β = "5": χ = "9"

This variable information is used with the Signature associated with the other Feature of the FeatureCouplet that contained the Signature that shared constant equivalence with the input utterance, to hypothesise a reply. Hypothesising a reply is processed by over-writing variable slots in the response Signature with variable elements deduced from the stimulus Signature, using the symbols contained in the response Signature's associated Feature as placement rules. It should be noted that it is only possible to hypothesise a full response provided that all the variable symbols in the response Feature appear in the stimulus Feature of the FeatureCouplet that contains the constant equivalence Signature. In the illustrative example this condition is satisfied as follows:

]β[plus]χ[is written as]β[+]χ[with α = "": β = "5": χ = "9"

produces:

"5 plus 9 is written as 5 + 9"

In this example, the input utterance shares constant equivalence with the Signature associated with a Feature formed from *stimulus* utterances in the training set. However, no such restriction is prescribed by the theory that underpins Baby. When processing more complex linguistic interaction, Baby may compute Signatures formed from *response* utterances of the training corpus to be members of the constant equivalence set, in which case the *stimulus* Feature is used to hypothesise the response.

FeatureCouplets can be thought of as generalised representations of conversational interaction. Whereas variable elements derived from novel conversational stimuli can be thought of as those components that specialise the generalised representation in the context of the novel response.

2.7.5 Emergent properties

The textual examples employed so far have been contrived to conveniently illustrate Baby's basic operation. This simplicity however, hides some important aspects of Baby behaviour that emerge in complex application of the model, as follows.

2.7.5.1 Ambiguous parses.

The arrangement of characters between two given Strings can be such that more than one parse exists that complies with all constant heuristics. For example in the two Strings:

Are dogs canine?

Are cats feline?

the largest constants to be identified are:

[Are]dogs can[ine?]

[Are]cats fel[ine?]

at this stage, "**dogs can**" and "**cats fel**" are processed to establish whether they contain constants. This instance examples two constant formations (each two characters in length) whose formations are mutually exclusive as follows:

[Are]dog[s]can[ine?]

[Are]cat[s]fel[ine?]

and

[Are]dogs [ca]n[ine?]

[Are][ca]ts fel[ine?]

Assigning both constants in the same Signature, would result in a contradiction of constant heuristic 5. Where such cases occur, both the FeatureCouplets derived from such constant stings are retained in the database of FeatureCouplets.

The following emergent properties are described in conceptual terms without illustration. This is because the associated examples are textually extended and unwieldy. Examples of such properties appear later in the thesis when more complex language interactions are described and exampled.

It can be imagined that when the pattern extraction process is applied to large and varied training corpora, then many more than one FeatureCouplet may be formed. Such multi-FeatureCouplet sets give rise to emergent Baby behaviour as follows:

2.7.5.2 Meta-Features.

If two or more FeatureCouplets are created from a given training set, then the pattern extraction process attempts to detect Signatures within the Signatures associated with these FeatureCouplets. If they exist, then the derived Features-of-Features are referred to as meta(1)-Features. If two or more meta(1)-Features are formed, then the pattern extraction process attempts to identify meta(2)-Features. This process iterates upon each level of meta-Feature until no more levels of meta-Feature exist. This process also manifests the model component *forming hierarchical structures of patterns of data.* A negative correlation exists between meta level number and number of meta-Features at that level, consequently the number of meta levels is self-limiting. In research so far, no more than five levels of meta-Features have been detected even from corpora containing 1000 or more utterance pairs.

2.7.5.3 Spreading pattern activation.

When conversationally interactive processing is invoked for a novel utterance, where multi-Feature/meta-Feature sets are present, then the *activation* model component is externalised during conversationally interactive processing as follows. When Baby detects constant equivalence between an input stimulus and a Signature associated with one of the Features of a FeatureCouplet, then that FeatureCouplet is referred to as a PrimaryActivatedCouplet. The Feature of a PrimaryActivatedCouplet whose associated Signature exhibited constant equivalence with the input utterance is referred to as the CoupletMother, and the other Feature of the PrimaryActivatedCouplet is referred to as the CoupletDaughter. Following the detection of PrimaryActivatedCouplets, Baby initialises a spreading activation process upon the FeatureCouplet database as follows. Baby detects all FeatureCouplets containing a Feature identical to the DaughterCouplet of PrimaryActivatedCouplets. This has the effect of activating all potential response Features, not just the particular one associated with the Feature whose associated Signature was directly stimulated by constant equivalence.

2.7.5.4 Competitive responses.

When conversationally interactive processing is invoked for a novel utterance, where multi-Feature/meta-Feature sets are present, then whether the membership function of the resemblance set

is crisp or fuzzy, the resemblance set is typically not the singleton (between 2 and 40 have been observed in research so far). The reason for this is twofold.

1. Although each Feature is unique in the database of Features, different Features may be derived from the same Signature. The difference in the Features exists, not in their constant parts, but in the combination and selection of variable symbols. This means that more than one Feature may share constant equivalence with an input utterance because the Features share an identical Signature. In such cases each instance of constant equivalence spawns a competitive hypothesised response.

2. Each SecondaryActivatedCouplet spawns a competitive hypothesised response.

Where Baby produces more than one hypothesised response, members of the hypothesised response set are referred to as competitive responses. The competitive response set may contain more than one instance of a particular response and there may be more than one set of identical responses. In such cases, Baby invokes *COTLPin* by enumerating the size of each set of identical response and hypothesises the response belonging to the largest set. In the case where there is more than one largest response set, then an ambiguity exists. So far, this problem has not been resolved, members from each largest response set being considered equally likely. In practice, the hypothesised response is taken from the first of the largest response sets to be detected. There is no Baby-theoretic underpinning for this action.

2.7.5.5 Potential for wide domain language processing.

The underpinning theory, does not model a conventional concept of the nature of language. Consequently, models of the theory do not contain artefacts such as lexicons, re-write rules or semantic representations, for example. Potentially, the language domain that externalisations of the theory might process, is a function of the language to which they are exposed. Rather than conceiving a theory that explains the complexity of natural language, the underpinning theory allows the complexity of the domain itself to provide a solution. If after much more research than is described in this thesis, it is found that the underpinning theory is tractable, then it is hypothesised that wide domain capabilities would automatically follow.

2.7.6 Summary of Baby theory.

Following a short discussion concerning the philosophical plausibility of this research, this section has outlined the principles involved behind the externalisation of a theory that hypothesises the

simulation of the production and comprehension of natural language. A rigorous account of the model and its implementation is presented in chapter *4.0 NonfuzzBabe*. The main points are listed below along with some important implicit issues that emerge from the model as follows:

1. Baby utilises components that model some aspects of five phenomena that are observed both in human cognition and/or AI machines:

    - attention;
    - forming hierarchical structures of patterns of data;
    - pattern recognition;
    - activation;
    - reasoning with uncertainty.

2. Baby directly inputs, processes and outputs textual representations of language.
3. Baby is not restricted to a particular language domain, excepting the single constraint that the language must by represented in textual format.
4. This is a small constraint for an NLP system and even then, is self-imposed to simplify the data set.
5. The underpinning theory is independent of the style of language used, the topics addressed or indeed (as will be shown), in many cases, the native language employed.
6. Baby makes many decisions based upon *COTLPin*.
7. A crucial aspect of this approach, is that the language behaviour hypothesised is achieved without the use of conventional theories of the nature of language, e.g. lexicons, grammars, or other hand-crafted language modelling devices. Baby is configured to enable it to learn about the patterns of commonality in linguistic discourse, not to model human theories of the nature of language.
8. Finally, because the underpinning theory uses the complexity of natural language itself to solve the problem of processing it, wide domain capabilities are hypothesised.

In addition to these points, an important issue was raised in this section concerning the method by which Baby maps the input utterance to most closely corresponding features. In this explanation the correspondence algorithm used to simply illustrate the processes involved in Baby was that of constant equivalence. Membership of the constant equivalence set is crisp. It was mentioned, however, that when using other correspondence metrics, such set membership may well be fuzzy. It is under such conditions that Baby utilises the model component reasoning with uncertainty. Part two of this chapter deals with these issues.

2.8 Related research.

The processing of natural language by machine using the method employed by Baby appears novel. No research has been found that emulates Baby exactly. There is however, a growing group within the NLP community that employs methods of sufficient similarity to those used by Baby, for an interesting comparison to drawn.

2.8.1 Data-oriented language processing.

Data-Oriented Language Processing (DOLP), is a new approach to the language processing problem; one of the first to postulate the approach was Scha (1990). One of its major applications is that of Data-Oriented Parsing (DOP), also variously known as corpus-based interpretation and tree-bank grammar, (cf. van den Berg, Bod & Scha 1996c; Bod 1992 & 1996a & 1996b; Bonnema 1996; Charniak 1996a/b; Goodman 1996; Kaplan 1996; Rajman 1995a/b; Scha 1992; Sekine & Grishman 1995; Sima'an, Bod, Krauwer & Scha 1994; Tugwell 1995). Though proponents of DOLP share many of the basic principles associated with Baby - notably an interpretation of the theory that underpins Baby - their application of the theory differs substantially from its application in Baby. Proponents of DOLP do, however, allow for principled extensions to their approach. Baby fits with the DOLP approach using some of these extensions.

2.8.1.1 The DOLP hypothesis.

Bod and Scha (1996c), state:

> *Data-oriented models of language processing embody the assumption that human language perception and production works with representations of concrete past language experiences rather than with abstract grammar rules. Such models therefore maintain large corpora of linguistic representations of previously occurring utterances. When processing a new input utterance, analyses of this utterance are constructed by combining fragments from the corpus; the occurrence frequency of the fragments are used to estimate which analysis is the most probable one.*

This hypothesis shares several important concepts with the hypothesis that underpins Baby, namely:

- storing representations of concrete instances of past language experience;
- using fragments of stored representations to hypothesise new language;
- using the metric *most frequent* to evaluate the probability that a particular previous language will be successful in processing novel language.

2.8.1.2 The DOLP model.

In brief, the DOLP model is one that may be characterised as follows (précised from Bod & Scha (1996c)). Adult language users have experienced many utterances, each of which contain many constructions and ambiguous interpretations. The question is; which one of these constructions should be used to process a novel utterance? Stochastic grammars generally assume that the smallest constructions are the only relevant ones (competence grammars), even though this is known to be false. DOLP adopts a framework that does not prejudge this matter. Instead, it accepts that a person's sentence analysis process is in some way determined by their past language experience and these data are employed in an "unmediated way" to disambiguate multi-constructions. This is done by constructing novel utterances from fragments of analyses that occur in representations of previous language experience. The metric employed to determine which fragments are used and how they are combined to form an appropriate novel utterance, is that of occurrence frequency in the experience corpus. This provides the most probable analysis of a novel utterance based upon successful language interaction in the past.

Bod (1995) sets out conditions that specify instances of DOLP. According to Bod, models of language comprehension and production that contain components described by the following list, lay within the DOLP framework:

1. a definition of a *formal representation for utterances-analyses*,
2. a definition of the *fragments* of utterance-analyses that may be used as units in constructing new utterance,
3. a definition of the *operations* that may be used in combining fragments,
4. a definition of the way in which *the probability of an analysis* of a new utterance is computed on the basis of the occurrence-frequencies of the fragments in the corpus.

Based upon these requirements, Baby is a member of the data-oriented processing framework. The conditions as they apply to Baby are as follows:

1. text;
2. constants;
3. features;
4. *COTLPin*.

2.8.2 A comparison between Baby and DOP.

DOP applications are used here as a generalised exemplar of DOLP because DOP is the main application of DOLP in the literature. The main difference between DOP, as practised by Bod along with many other practitioners, e.g. Charniak (1996a & 1996b), Goodman (1996), Kaplan (1996), Sekine and Grishman (1995), Sima'an (1995 & 1996), Tugwell (1995), van den Berg et al. (1994) and this research, is in the method used to represent concrete instances of past language experience.

The DOP method uses Chomskyean type phrase structures employing syntactic function labels such as noun, verb or adjective etc., to represent language experience. In Baby models, concrete fragmented textual representations of language are employed. As far as can be established, no other published research has chosen this approach. The difference is so important that Baby cannot be considered a part of main stream DOLP as represented by DOP. An argument for inclusion within current DOLP work is not made, rather it is considered that Baby externalises the shared underpinning theory in a way that may provide more versatile language processing. This is not to say that main stream DOLP practitioners are dogmatic about the particular representation of language that they have chosen. Bod and Scha (1996c) for example allow for other formalisms by stating:

> *We hypothesise that human language processing can be modelled as a probabilistic process that operates on a corpus of representations of past language experiences, but we leave open how the utterance-analyses in the corpus are represented, what sub-structures of these utterances-analyses play a role in processing new input, and what the details of the probabilistic calculations are.*

Indeed, in the same paper they appear, in some respects, to anticipate Baby by saying:

> *If we want to avoid stipulating an "innate grammar", we must show how a corpus with syntactic structures may gradually come into being, starting with an initial state in which a child only has a corpus with non-linguistic experiences. ..... Describing this process* [representing semantics and pragmatics] *in any great detail will obviously be very difficult. Nevertheless, it constitutes a more promising research agenda than trying to find a connection between the observable child language acquisition data and the process of setting parameter values in an innate super-grammar.*

2.8.3 Summary of related research.

In this section it has been implied that the Baby model of NLP, (see Lefley & Rogers, 1997; Lefley & Rogers, 1998; Lefley & Rogers, 1998a; Lefley & Rogers, 1999; Rogers & Lefley, 1997; Rogers & Lefley, 1997a; Rogers & Lefley, 1999), is probably unique amongst published NLP applications. Consequently the research described in this thesis cannot be said to be a direct extension of previous work in language processing. One relatively new avenue of NLP research, that of Data-Oriented Language Processing (DOLP), does however, assume an underpinning theory that is similar in some important ways with the theory that underpins Baby. Most of the work that utilises the DOLP approach is concentrated in the area of Data-Oriented Parsing DOP. DOP was compared with Baby and the issue of the different methods employed by each model, of representing past language experience was highlighted. The difference was seen to be sufficiently significant to position Baby outside the main stream of DOP, (and thus DOLP) research. It was noted however, that DOLP practitioners accept shortcomings in their externalising model and allow principled extensions to be researched. It was shown that Baby can be said to fall within the DOLP framework assuming two such extensions is pursued, that of employing different language representation formalisms and probability calculation.

Part Two: Reasoning with uncertainty.

2.9 Introduction.

It can be seen from many examples in the contents of part one of this chapter, (e.g. *2.6.2 Extending SEARS to Baby.* and *2.7.2.4 Reasoning with uncertainty*) that in order for Baby to adequately model the fuzzy aspects of Dr. Lefley's original aims (see *2.6.1 SEARS.*) it was necessary to address the issue of representing and processing uncertain knowledge. Part two of this chapter covers these issues. Firstly, many of the more popular formalisms are briefly described. This is followed by the reasons for choosing the formalism and how it is implemented in Baby models.

2.10 Formalisms.

It is necessary for Baby to reason with uncertainty in order to model the concept represented by the words "most closely correspond" in its underpinning theory. Configuring machines to solve problems within non-specified or dynamic problem domains form a significant part of AI research.

There follows a brief discussion of some of the formalisms available for dealing with uncertainty, along with their advantages and disadvantages. Subjective and objective interpretations of belief are not discussed here, see Krause & Clark, (1993) for a good account. This chapter assumes a subjective perspective.

2.10.1 Bayesian models.

In these models, a mathematical definition of probability theory is employed to establish the truth of a given hypothesis in uncertain domains. An important goal for problem solving systems is to assemble evidence as the process proceeds and to modify the probability of given candidate solutions accordingly. The application of the Bayesian formalism provides for this by enabling the evaluation of the probability of hypotheses in the light of new evidence (posterior probability), by considering:

- the probability assigned to the hypothesis without the new evidence (prior probability);
- the probability of the new evidence given that the hypothesis is true (conditional probability).

Cheeseman gives a concise narrative definition for a subjective interpretation of Bayes as follows:

The (conditional) probability of a proposition given particular evidence is a real number between zero and one, that is an entity's belief in that proposition, given the evidence. (Cheeseman, 1985)

Advantages.

Perhaps the main advantage of the Bayesian approach is that it imposes a strict mathematical discipline on knowledge engineering under certain well-defined conditions. Given that all the information required for substitution into the Bayes formula is available and accurate, then the calculated value of the posterior probability of an hypothesis for a given set of conditions is also accurate.

Disadvantages.

If the well-defined conditions mentioned above are not met, then applying a Bayesian approach may impose some difficulties. Firstly, probability theory can be used to navigate uncertain domains, only if the uncertainty relates to atomic or proposition conditions. Consequently, concepts such as inconsistency, incompleteness and irrelevance are not allowed. Secondly, it is difficult to evaluate the prior probability of hypotheses and conditional probability of evidence. One reason for this is that it has been shown that people are very poor at estimating such data, e.g. Tversky & Kahneman, (1974) or Kahneman, Slovic & Tversky (1982). Thirdly, the number of probabilities required to model a given domain increases exponentially with increasing domain size, thus attracting computational intractability in scaling efforts.

2.10.2 Certainty factor models.

Certainty factor (CF) formalisms were developed, in part, to overcome some of the problems associated with applying the Bayesian formalism without proper consideration of the required conditions. Such inappropriate applications were referred to as 'idiot Bayes' (Duda, Gashnig & Hart 1979). CF models attempt to cater for incomplete information by:

- allowing belief in given hypothesises to be updated without necessarily decreasing belief in other related hypotheses;
- allowing the addition or deletion of new evidence to occur without the belief in other evidence being effected.

Advantages.

The main advantage with the CF formalism is that it allows heuristic reasoning to be encoded as deductive rules, whilst allowing uncertainty to be modelled within a formal but syntactically simple

calculus. Furthermore, CF has been successfully employed in expert systems, e.g. Prospector (Duda et al, 1979)

Disadvantages.

The CF formalism allows irrelevant knowledge to be added to an existing knowledge base. This is useful because such knowledge may become relevant later. The price to be paid for this is that because CF therefore assumes semantic modularity, it renders the formalism limited in its expressiveness and confines use to highly circumscribed conditions (Krause & Clark, 1993).

2.10.3 Belief functions.

The Dempster-Shafer (D-S) theory (Dempster, 1968; Shafer, 1976) forms a generalisation of the Bayesian model. Instead of considering individual propositions and assigning to them single number estimates, as with previous formalisms described so far here, D-S considers sets of propositions and assigns to each set a belief factor.

Advantages.

One advantage of D-S is that, with only a slight relaxation of axioms, beliefs need not be additive. This means that D-S allows for the probability of the truth of a belief and the probability of a belief being false not necessarily summing to 1. Such a condition can occur when people estimate the probability for a given belief. Another advantage is that D-S allows for explicit statements of ignorance, and degrees of inconsistency between sources of evidence. Finally, D-S allows for the pooling of evidence from a variety of sources.

Disadvantages.

The general nature of D-S however, attracts some disadvantages. The main problem is that the problem rapidly becomes computationally intractable as the number of sources of evidence increases, i.e. the problem becomes computationally exponential (Orponen, 1990). It should be noted however, that this problem has been partially overcome by the use of approximation techniques (e.g. Barnett, 1981; Shenoy & Shafer, 1986; Shafer, Shenoy & Mellouli 1987; Shenoy & Shafer, 1990; Xu, 1991).

2.10.4 Possibility theory models.

Possibility theory and fuzzy logic provide the principal formal systems explicitly devoted to the representation and manipulation of incomplete knowledge manifested as vagueness (Krause & Clark, 1993). A numerical calculus, that represents partial knowledge (referred to as likelihood, propensity

or confidence), is combined with a set theoretic component that represents uncertain knowledge or vagueness.

Advantages.

Data expressed as fuzzy sets allows for the representation of ambiguity because overlaps are allowed between membership functions of different concepts. In addition, partial degrees of inconsistency, incompleteness, ignorance and irrelevance can be modelled by this method (Rich & Knight, 1991). Overall, possibility theory "*continues to be a rich area of research that has spawned many applications and has a clear role in the domain of uncertain reasoning*" (Krause & Clark, 1993).

Disadvantages.

Possibility theory is less tightly constrained than the other formalisms discussed so far in this chapter and may therefore exhibit overproductivity. Also, uncertainty distributions produced are less precise (though this does have computational advantages). In the case of fuzzy logic, possibility distributions typically extend across a range of values and this has the disadvantage of increasing the difficulty in obtaining reliable estimations from human sources.

2.10.5 Non-monotonic models.

Non-monotonic models enable the understanding of inference patterns, that deal with incompleteness, by making problem specific assumptions. Different approaches have been employed to achieve this goal including:

- minimalist reasoning and closed world assumptions theories, e.g. Reiter (1978), and theories of circumscription, e.g. McCarthy (1980); McCarthy (1986) and Lifschitz, (1985);
- default reasoning such as nonmonotonic logic (McDermott & Doyle, 1980), default logic (Reiter, 1980), abduction, and inheritance.

Advantages.

Minimalist reasoning systems work well in special purpose systems because of their reliance upon closed world assumptions. Examples of special purpose systems include:

- databases where there are no more instances of a relation than those deducible from the database itself (Reiter, 1984);
- inheritance networks, where general classes are assumed to inherit the properties of their superclass, unless this conflicts with some more specific information (Touretzky, 1984);
- truth / reasoning maintenance systems;
- logic programmes employing negation as failure (Clark, 1978).

Default reasoning systems rely on extensions of classical logic and can be applied to more general domains than minimalist systems can handle.

Disadvantages.

Minimalist formalisms obviously cannot handle domains that cannot be closed, which accounts for many human behaviour domains. Though challenging, default reasoning systems have not proved to be a comprehensive, satisfactory formalism and consequently none have been successfully demonstrated in large real domains (Krause & Clark 1993).

2.10.6 Other formalisms:

For the sake of completeness, other formalisms are listed below that are considered either irrelevant to this research, or share much in common with other formalisms already described. Intentionally, the last three are referred to by citation only.

2.10.6.1 Argumentation.

A central component of intelligence is the ability to understand and engage in argument. Argumentation has been considered for a relatively short period by researchers as a tool for navigating uncertain knowledge spaces and little established work has been published, see Alvarado (1992) for a good general review of the subject.

2.10.6.2 Verbal uncertainty expressions.

The parameters required by the various formalisms described so far can be evaluated directly from data. In cases where such parameters can only be elicited from human experts, it is necessary to evaluate the expert's confidence in particular propositions from their natural language expressions of uncertainty about the propositions. Such verbal expressions are rated by experts, usually in a normalised interval in the range 0-1. Information thus gained is used to develop fuzzy set membership functions. The reader is referred to Clark, (1990) for a good overview and Clark, (1988); Fox, (1986) and Bonissone, (1992) for accounts of typical implementations.

2.10.6.3 RUM-PRIMO.

See Bonissone, Gans & Decker (1987)

2.10.6.4 Upper and lower probability bounds.

See Dubois & Prade, (1982)

2.10.6.5 Qualitative uncertainty.

See Parsons, (1992); Parsons & Mamdani, (1993) or Wellman, (1990)

2.10.7 Analysis of formalisms capable of processing nonmonotonic problem spaces.

Given that the above list attempts to represent an exhaustive taxonomy of the main formalisms capable of reasoning with uncertainty, each one was inspected to establish a contender most suited to the tasks required by Baby. Table 1. summaries the findings:

Formalism	Main advantage	Main disadvantage	Suitability for Baby
Bayes	High accuracy	Requires accurate prior condition information	Cannot assess prior probability information
Certainty factor	Allows heuristic rules to be treated as deductive rules.	Computationally intractable for large state spaces. Requires certainty factor estimates	Cannot assess certainty factors
Possibility theory	Allows representation of ambiguity	Possibly inaccurate and/or over-productive	Tractable
Nonmonotonic models	Relies on extension to well proven formal logic	Requires closed world assumptions	Baby concept is open world

Table 2: Summary of nonmonotonic formalisms for application in Baby models.

2.10.8 Summary of formalisms for reasoning with uncertainty.

Of the formalisms considered, it can be seen that, by default, possibility theory is the most likely contender. This is because, of the formalisms discussed, it is the only one capable of representing problems in an open world domain where it is not possible to accurately assign values to either prior probability or confidence.

It is envisaged that possibility theory will be implemented by representing the correspondence between a given input utterance and a given feature by a fuzzy indicator. This indicator would take a value in the interval [0 - 1], where 0 would represent zero correspondence, 1 would represent an identical match and values in-between would represent degrees of correspondence. Such values may be directly calculated from the data associated with the input utterance and each feature. That is,

correspondence may be expressed as some function of the length of character Strings that are common between an input utterance and a feature's associated signature.

This method has the advantage that the fuzzy indicator does not rely upon hand-crafted estimates of confidence or prior probability. However, the disadvantage, as highlighted in table 1, i.e. inaccuracy and overproductivity, may render the formalism useless for this application. This dilemma and its resolution are addressed in chapter *3.0 Research strategy*.

2.11 Fuzzy logic.

It was not considered appropriate to devote effort to a detailed exposition on the theory and application of fuzzy processing techniques, due to the size and complexity of the subject. Instead, a brief résumé of those issues relevant to fuzzy implementations of Baby is given here.

2.11.1 Background.

Real world problems are often complex. This complexity arises from uncertainty that generally stems from ambiguity. Problems that feature uncertainty and ambiguity are commonly solved by humans, though computer implementations have traditionally found them difficult. One basic reason for this difference in problem solving performance is that "*humans have the capacity to reason approximately*" (Ross, 1995), thus allowing them to form a generic understanding of the world. Dr. Lotfi Zadeh (1972) presented a fuzzy formalism that enabled the representation of problems by approximating problem states using fuzzy sets. Zadeh theorised that computers implemented to process fuzzy sets would emulate the approximation that humans use in their reasoning.

2.11.2 Underpinning theory.

In Fuzzy set theory, a domain that subsumes all the available information on a given problem is referred to as the problem's *universe of discourse*. Once this universe is defined, then issues relating to the problem may be represented by sub-domains within the universe of discourse. Mathematical abstractions of such sub-domains are referred to as sets. Classically, sets had been abstracted to possess crisp boundary conditions, i.e. set membership is either true or false, 0 or 1. The characteristic function of a crisp set A comprising elements x is:

$$\chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

Fuzzy sets are described by vague or ambiguous properties and set membership is represented by degrees of likelihood. Likelihood is usually expressed by a fuzzy indicator that takes a value on the interval [0 - 1], i.e. the membership function of a set B comprising elements y is:

$$\mu_B(y) = \begin{cases} 0\text{-}1, & y \in B \\ 0>y>1, & y \notin B \end{cases}$$

2.11.3 Implementation of fuzzy controllers.

Simple fuzzy controllers are normally configured in a similar manner. Figure 2 is a directed graph that illustrates the main processes involved, (from Passino & Yurkovich, 1996):

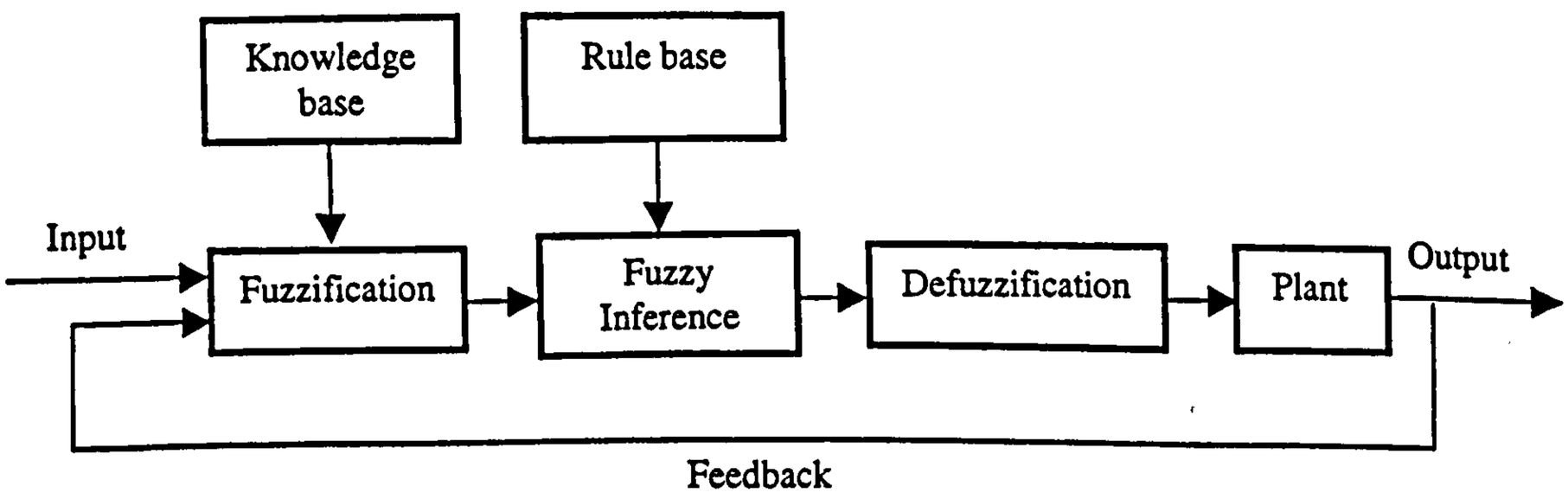


Figure 2: The main processes involved in a typical fuzzy controller.

2.11.3.1 Fuzzification.

Fuzzification is the process of making crisp quantities fuzzy. If a crisp value is to represent the solution to a problem that contains imprecision, ambiguity or vagueness, then the crisp value must be mapped to a fuzzy value that indicates the likelihood of solution being correct. The mapping function between crisp values and likelihood of them being members of a given fuzzy set is called the membership function of the fuzzy set. In Figure 2, the fuzzification process takes input and a knowledge base and produces a set (possibly the singleton) of fuzzy indicators that define the likelihood of input being a member of each piece of knowledge. The fuzzification process may use several metrics to establish various fuzzy values that indicate membership for each piece of knowledge.

2.11.3.2 Fuzzy inference.

The fuzzy inference process takes as input the set of fuzzy indicators evaluated during fuzzification and a set of hand-crafted rules, and outputs a single fuzzy value that indicates a fuzzy solution to the current problem as modelled by the rule set. Most rules take the general form:

IF *condition* THEN *action*

For multi-conditional actions, conditions may be combined (e.g. logical ANDing and ORing). For example the general form for a two condition action may be written:

IF *condition1* AND *condition2* THEN *action*

or

IF *condition1* OR *condition2* THEN *action*

Two general methods of fuzzy inference model the **AND** and **OR** connectives in fuzzy rule bases:

1. **OR** – fuzzy max-min composition:
   Given that R is a fuzzy relation on X x Y, S on Y x Z and T on X x Z then fuzzy max-min composition is defined as:

$$T = R \circ S$$

$$\mu_T(x, z) = \text{Max}[\text{Min}[\mu_R(x, y), \mu_S(x, z)]]$$

2. **AND** – fuzzy max-product (sometimes referred to as fuzzy max-dot) composition:
   Given the same set theoretic domain as fuzzy max-min, then fuzzy max-product composition may be defined as:

$$\mu_T(x, z) = \text{Max}[\mu_R(x, y) \bullet \mu_S(x, z)]$$

2.11.3.3 Defuzzification

Defuzzification is the process of making fuzzy quantities crisp. According to Hellendoorn and Thomas (1993), there are at least seven defuzzification strategies:

- max membership principle;
- centroid method;

- weighted average method;
- mean-max membership;
- centre of sums;
- centre of largest area;
- first (or last) of maxima.

The max-membership principle is used in Baby and therefore discussion is confined in this sub-section to defining this method. The method is also known as the height method and is limited to peak output functions. The method may be represented by the algebraic expression:

$$\mu_C(z^*) \geq \mu_C(z)$$

where:

C is a set of fuzzy indicators z;

z^* is the sub-set of C of maximum values of z.

A discussion on the relative merits of each of these defuzzification methods is outside the scope of this thesis, (but see Ross, 1995 for a good account).

2.12 Overall Chapter Summary.

The chapter began with an overview of the domain within which Baby research was conducted. The overview included a high level description of the methodologies available, along with theoretical factors that have limited the performance of applications to date. It was noted that Baby research may allow the possibility of overcoming some aspects of the limitations associated with machines that acquire problem solving behaviour during exposure to large and complex problem spaces. This was followed by an overview of machine learning strategies used in artificially intelligent research, and a positioning of Baby within these strategies. The remainder of the chapter was divided into two parts, each dealt with two related, important and distinct research disciplines. Part one dealt with issues relating to NLP, and part two dealt with issues relating to reasoning with uncertainty.

Part one, began with the reasons for choosing NLP as a test domain. An introduction to NLP was then given and problems associated with wide domain processing were highlighted. An overview of previous NLP research was given and this was followed by example applications. The predominance of this overview was devoted to language acquisition machines. Baby research was then explained and a detailed account was provided of how it significantly extended antecedent research. The Baby model was then superficially described. There followed an exposition of related

work by other researchers, and an argument was given for how this research significantly differed from that work.

Part two began with a review of current formalisms employed in designing machines that reason in incomplete problem domains. Formalisms were reviewed with respect to Baby research and an argument was provided for the reasons for choosing possibility theory (manifest in fuzzy logic) for this research. Fuzzy logic was then briefly described. The chapter concluded with a summary of both parts one and two.

Chapter 2 – Domain, grounding and motivation.

3.0 Research strategy.

The research domain circumscribed by Baby, as described so far in this thesis, is too large and complex to be investigated adequately and rigorously enough, within the time scale and resource available for this research project. It is necessary, therefore, to reduce the research domain by selecting one from many contending central issues connected with the underpinning theory. The following criteria were applied when considering contending issues. The research should:

- be tractable;
- provide experimental data that tests an aspect of the theory underpinning Baby;
- be of a size commensurate with part of a doctoral research programme;
- provide a foundation for further research into Baby;
- contribute to other related research;
- contribute to human knowledge in general.

3.1 Research domain.

A discussion in *2.10.4 Possibility theory models.*, introduced the difficulties that may be faced by employing fuzzy classification techniques to model the concept represented by the words "most closely correspond" in Baby's underlying theory. The concept of correspondence is an essential element of the theory. If, for instance, the words "identically match" were substituted (and the following "with" omitted), then the research described here may not have been conducted. This is because it is thought likely that such a revised proposition, probably theorises no more than trivial human language behaviour. In addition, such a theory could be argued as behaviourist in nature[1].

---

[1] A personal excursion.

The community began rejecting the behaviourist paradigm in favour of cognitive science in the 1950s, with the now infamous Chomsky vs. Skinner debate, (Chomsky, 1965; Skinner, 1957). The opinion presented here, is that such a total rejection of behaviourism may have resulted in thowing Baby out with the bath water, so to speak. It is believed that the *deterministic* relationship between stimulus and response is behaviourism's Achilles heel, rather than any theoretical intractability in the underlying concept. Rather than shifting to the diametrically opposed paradigm of cognitive science, consideration of a *fuzzy* 'black box' may also have yielded some important results. It can be imagined that Baby is, in fact, an externalisation of exactly such a concept. From a personal perspective, testing this notion has provided a major motivation for the work.

Notwithstanding this state of affairs, it was considered sensible to test the Baby model using identical match classification in the first instance, in order to explore the following issues:

- whether Baby could process language at any level, and if so;
- indicate a reference language performance, against which Baby implementations using fuzzy classification techniques could be compared.

If the exploration produced encouraging results, then it would provide confidence to investigate the effect of employing fuzzy classification techniques to model the concept 'most closely correspond' in Baby's underpinning theory.

It was shown in section *2.10.7 Analysis of formalisms capable of processing nonmonotonic problem spaces.*, that possibility theory allows the automatic calculation of fuzzy indicators to model correspondence in Baby. Other formalisms require either closed world assumptions, or hand-crafted estimates of confidence or prior probability and this made them unsuitable for use with Baby. On the other hand, possibility theory may exhibit the disadvantage of inaccuracy and/or overproductivity.

The domain to which the research in this thesis relates, constitutes a combination of the above two sub-domains, i.e.:

- to investigate the language competence of Baby implemented with exact match classification techniques;
- investigate the effect of fuzzy classification techniques on the language competence of Baby.

3.2 Experimental hypotheses.

Identical match classification is generally known as 'equivalence classification', e.g. Terano, Asai & Sugeno (1994), or Nauck, Klawonn & Kruse (1997), or Jang, Sun & Mizutani (1997), and this convention will be followed here. Given this, the above two sub-domains of this research may be formulated as two experimental hypotheses.

3.2.1 Hypothesis one

It is hypothesised that Baby models, implemented with equivalence classification techniques will, following a pattern extraction process on a representative corpus, be capable of hypothesising correct responses to simple, novel textual representations of conversationally interactive stimuli, as judged by a human user of the language employed.

3.2.2 Hypothesis two

It is hypothesised that Baby models, implemented with fuzzy classification techniques, will produce superior simple language processing performance compared with Baby models implemented using equivalence classification techniques, as judged by a human user of the language employed.

It can be seen that the testing of hypothesis two is dependent upon experimental evidence that supports hypothesis one.

3.3 Experimental set up.

In order to test the experimental hypotheses, it was decided to implement three versions of the Baby model. A version using equivalence classification and two versions employing degrees of fuzzy classification. The models are described as follows:

- one using equivalence classification;
- one using limited fuzzy match classification;
- one using more extensive fuzzy match classification.

The names chosen for each of these implementations were NonfuzzBabe, CorrBabe and FuzzBabe, respectively (abbreviated forms of non-fuzzy Baby, correspondence Baby and fuzzy Baby). A specification for the pattern classification algorithm for each implementation follows.

Given:

Two sets A and B on a universe of discourse of Strings Z;

Two sets D and E on a universe of discourse of Features Y;

Where:

$\beta \in B$ (a singleton);

D = the set of Features derived from A;

$\varepsilon \in E$ = members of D that correspond with β;

$A \cap B$ = the null set ϕ;

$E \subseteq D$.

Then:

3.3.1 Equivalence classification (NonfuzzBabe).

The characteristic function of set E [$\chi_E(\varepsilon)$] in NonfuzzBabe may be stated:

A Feature $\varepsilon \in E$ iff ***all of each*** constant in the SignaturePart the Feature ε exist in the same order, left to right, in β.

e.g. equivalence classification exists between:

"*Where is your* cricket b*at?*" and **][Where is your][at?][**

and does not exist between:

"*Where is your* cricket ball?" and **][Where is your][at?][**

3.3.2 Limited fuzzy match classification (CorrBabe).

The membership function of set E [$\mu_E(\varepsilon)$] in CorrBabe may be stated:

A Feature $\varepsilon \in E$ iff ***a concatenated element that shares the two left-most characters of EACH*** constant in the SignaturePart of the Feature ε exist in the same order, left to right, in β.

e.g. limited fuzzy match classification exists between:

"*Where*'s your cricket hat" and **][Where is your][at?][**

and does not exist between:

"*Where*'s your cricket ball?" and **][Where is your][at?][**

3.3.3 More extensive fuzzy match classification (FuzzBabe)

The membership function of set E [$\mu_E(\varepsilon)$] in FuzzBabe may be stated:

A Feature $\varepsilon \in E$ iff ***a concatenated element that shares the two left-most characters of ANY*** constant in the SignaturePart of the Feature ε exist in the same order, left to right, in β.

e.g. more extensive fuzzy match classification exists between:

"*Where*'s your cricket b*at?*" and **][Where is your][at?][**

and also exists between:

"*Where*'s your cricket ball?" and **][Where is your][at?][**

and also exists between:

"There is your cricket b*at*." and **][Where is your][at?][**

but does not exist between:

"There is your cricket ball." and **][Where is your][at?][**

NonfuzzBabe will be used to test hypothesis one and all three versions will be used to test hypothesis two. It is proposed that the domain as described is likely to fulfil the criteria set out at the beginning of this chapter as follows:

- it is considered tractable;
- the data will test an aspect of the theory underlying Baby;
- it is commensurate with part of a five year research programme;
- research could continue into applying Baby to different language domains or investigating the effect of different fuzzy processing techniques;
- the findings may be of use to other DOLP research;
- the findings may be of use to practitioners of AI in general.

3.4 Summary.

In this chapter, a case was made that the domain circumscribed by Baby is too large to adequately researched for this project. A subset of the domain was chosen that complied with a set of self imposed criteria of excellence and two experimental hypotheses were developed. It was then shown how three versions of Baby could be used to gather experimental evidence to test these hypotheses.

Chapter 3 – Research strategy.

4.0 NonfuzzBabe.

In this chapter, the terms system, module and component, refer to an hierarchical structure with system at top level and component at bottom level. The chapter contains a detailed account of the NonfuzzBabe system by addressing:

- a review of system rationale;
- definitions of terms used;
- overall system behaviour;
- a specification for each of the modules that comprise the system;
- a process flow representation for each module of the system;
- component input/output specification;
- a system data flow diagram;
- a system data dictionary.

N.B. Reference is made in this chapter to the concept of Variable in the guise of VariableCluster, VariablePart and Variable Extractor. **NonfuzzBabe does not use any of these constructs in its processing of response to novel utterance.** The concepts are defined, and processes involving their detection and storage only, are described in this chapter. Variables form a major part of chapter *8.0 Illustrations of advanced behaviour.*, where versions of Baby are described that use fuzzy Variable indicators as part of processing response to novel utterance. The decision to introduce them in this chapter was taken in order to avoid excessive duplication of text in Chapter 11.

This chapter concludes with a summary of the main points covered.

4.1 Rationale.

NonfuzzBabe was designed, implemented and tested in order to gather data that could be used to test the first hypothesis relating to this research project (see section *3.2.1 Hypothesis one*).

If the data gained supports this hypothesis, then this data will also be used as a base language performance metric, with which fuzzy versions of Baby will be compared, in order to test the second hypothesis stated in section *3.2.2 Hypothesis two.* NonfuzzBabe is designed to model equivalence classification. A specification of the membership function for NonfuzzBabe's classification set is shown in section *3.3.1 Equivalence classification (NonfuzzBabe).*

Accounts of research into NonfuzzBabe (under the guise Babette) have been published in Rogers & Lefley (1997a), Rogers & Lefley (1997c), Lefley & Rogers (1998a), Lefley & Rogers (1998b) and Rogers & Lefley (1999).

Designing, implementing and testing a system that detects the largest textual pattern frequencies contained in a training corpus, and employing such patterns to process responses to novel utterances, appears relatively straight forward. Research into NonfuzzBabe however, exposed unforeseen complexities. These complexities stem from the novel nature of the method, that necessitated a paradigmatic shift in conceptual understanding of computer language processing. The complexities are mostly concerned with the requirement to develop a new domain of concepts and abstract artefacts with which to model and implement the process. Also, ambiguous parses and the concepts MetaFeature and competitive response described in section *2.7 The Baby model.*, proved interestingly complex.

4.2 Definitions.

This section contains terms that are either coined, or have wider meanings in other fields of interest. A context dependent list of definition for such terms follows. Simple examples have been used to illustrate concepts in order to aid understanding. Where reference is made to the terms defined here, the same textual format is used as in definitions. For example "String" is as defined here, whereas "string" may refer to any of its dictionary definitions.

1. **Character:** any of the ASCII character set except 47 (/), 126 (~), 91([) and 93(]).
2. **String:** any combination of **Characters.**
3. **NullString:** an empty **String.**
4. **TextCouplet:** any two **Strings** that represent a conversational exchange within a discourse;

   e.g. **Where is your coat?** or **Where is your hat?**

   My coat is on the bed. **My hat is on the table.**
5. **StimulusTextCouplet:** the **String** of a **TextCouplet,** that occurs first in the discourse of which the **TextCouplet,** was a part.
6. **ResponseTextCouplet:** the **String** of a **TextCouplet,** that occurs second in the discourse of which the **TextCouplet,** was a part.
7. **TextCouple:** any pair of **StimulusTextCouplets** or any pair of **ResponseTextCouplets;**

   e.g. **Where is your coat?** or **My coat is on the bed.**

   Where is your hat? **My hat is on the table.**

Definitions 8 through 14 specify relationships between the two Strings of a TextCouple;
(brackets bound sub-Strings that example relationships)

8. **Common** element: a sub-**String** of one **String** of a **TextCouple** that is coincident in the other **String** of the **TextCouple;**

   e.g. **Wher[e is y]our coat?** or **My coat i[s on t]he bed.**
   Wher[e is y]our hat? **My hat i[s on t]he table.**

9. **Constant:** a **Common** element that complies a set of heuristics (see *2.7.4.1 The pattern extraction process.*);

   e.g. **[Where is your]coat?** or **My co[at is on the]bed.**
   [Where is your]hat? **My h[at is on the]table.**

10. **Variable:** an element of one **String** of a **TextCouple** that is not a **Constant;**

    e.g. **Where is your [co]at?** or **My coat is on the [bed].**
    Where is your [h]at? **My hat is on the [table].**

11. **ConstantString:** one **String** of a **TextCouple** with its **Variable** element(s) removed;

    e.g. **Where is your at?**

12. **Signature:** a **ConstantString** with its **Constant**(s) elements identified by brackets;

    e.g. **[Where is your][at?]**

13. **AnnotatedSignature:** a **Signature** with each of the **Variable** element location(s) labelled with an integer. Each integer symbolises **Variable** elements that are derived when a **ConstantString** is removed from **String**(s) containing that **ConstantString.** Such that for each **Variable** derived, a unique integer represents each different **Variable** element and that each identical **Variable** element is represented by the same unique integer. Integers are allocated in a sequentially ascending series from left to right of the **Signature** (unless they symbolise a **Variable** element identical to one that had occurred earlier in the **Signature**);

    e.g. for the **String:** **Where is your coat?**
    and the **ConstantString:** **Where is your at?**
    the derived **AnnotatedSignature** is: **]0[Where is your]1[at?]0[**
    where: **0** symbolises the **NullString** and **1** symbolises "co"

14. **VariableCluster:** the set of **Variable** element(s) obtained from the derivation of an **AnnotatedSignature;**

    e.g. where "/" delimit **Variable** elements, the **VariableCluster** derived between:
    the **String:** **Where is your coat?**
    the **AnnotatedSignature:** **]0[Where is your]1[at?]0[**
    is: **//co//**

Definitions 15 to 27 specify data structures:

15. **Feature:** a fourtuple data set comprising:
    a) a **ConstantString;**
    b) a **Signature;**
    c) an **AnnotatedSignature;**
    d) a database containing **VariableClusters** derived from **TextCouplets** that form the datum **c)** of the Feature.

        e.g. where the character "*/*" delimits each **Variable** in a **VariableCluster** and the character "**~**" delimits each **VariableCluster:**

        the following data (*a*, *b*, *c* and *d*) example a Feature with one *a*, *b* and *c* type datum and two *d* data:

        a) **Where is your at?**
        b) **][Where is your][at?][**
        c) **]0[Where is your]1[at?]0[**
        d) **//co//~//h//**

16. **ConstantPart:** that part of a **Feature** containing data type (*a*) in 15 above:
17. **SignaturePart:** that part of a **Feature** containing data type (*b*) in 15 above:
18. **FeaturePart:** that part of a Feature contained in data type (*c*) in 15 above:
19. **VariablePart:** that part of a **Feature** containing data type (*d*) in 15 above:
20. **FeatureCouplet:** a **TextCouplet** expressed as two **Features;**

    e.g. where a colon character separates each data part:

    Where is your at?:][Where is your][at?][:]0[Where is your]1[at?]0[://co//~//h//

    My at is on the .:][My][at is on the][.][:]0[My]1[at is on the]2[.]0[://co/bed//~//h/table//

21. **ConstantEquivalence:** exists when the ordered set of **Constant(s)** of the **SignaturePart** of a **Feature** occur sequentially in a **String.**

    e.g. Constant equivalence exist between:

    "Where is your cricket bat?" and **]0[Where is your]1[at?]0[**

    and does not exist between:

    "Where is your cricket ball?" and **]0[Where is your]1[at?]0[**

22. **StringCorrespondence:** the ratio of the number of **Characters** that comprise **Constant** element(s) and the floor of the average length of two **Strings;**

    e.g. **[Where is your]co[at?]**

    [Where is your]h[at?]

    StringCorrespondence = 17 / ⌊(37 / 2) = 0.944444

23. **DominantFeatureCouplet:** a FeatureCouplet derived from the set B where:

    A = a set of TextCouplets;

    $B \subseteq A$ = TextCouplets whose TextCouples share the largest StringCorrespondence.

24. **PrimaryActivatedCouplet:** a **FeatureCouplet** containing a **Feature** whose **SignaturePart** exhibits some correspondence with an input utterance.
25. **CoupletMother:** the **Feature** of a **PrimaryActivatedCouplet** whose **SignaturePart** exhibits some correspondence with an input utterance.
26. **CoupletDaughter:** the **Feature** of a **PrimaryActivatedCouplet** that is not the **CoupletMother**
27. **SecondaryActivatedCouplet:** a **FeatureCouplet** containing a **Feature** whose **SignaturePart** is identical to the **CoupletDaughter** of a **PrimaryActivatedCouplet** (i.e. the set of **PrimaryActivatedCouplets** is a proper subsets of the set of **SecondaryActivatedCouplets**).

Definitions 28 through 30 specify processes used by all Baby models including NonfuzzBabe:

28. **Feature Extraction:** the process of computing a list of **FeatureCouplet**(s) from at least two **TextCouplets.**
29. **MetaCouplets Extraction:** the process of computing a list of **MetaCouplet**(s) from at least two **ConstantStrings.**
30. **Pattern Extraction:** the combined process of **Feature Extraction** and **MetaCouplet Extraction.**

4.3 System description.

The following issues are addressed in order to describe the NonfuzzBabe system:

- pre-conditions;
- overall system behaviour produced by the interaction of various modules;
- the operation of each of these modules, with a process flow diagram;
- a directed graph representation of the complete system in the form of a data flow diagram;
- a data dictionary.

4.3.1 Pre-conditions.

The following list of pre-conditions are assumed in the context of system behaviour, module and component descriptions:

1. the human interlocutor referred to, is one who is a competent user of the language to which NonfuzzBabe is being applied;
2. conversational utterances are input and output using textual representation;
3. during a conversation, utterances alternate between the human and NonfuzzBabe, with the human providing the first utterance and NonfuzzBabe providing the last;
4. the term *current utterance* refers to the utterance most recently produced as part of the current conversation. The positions of other utterances in the conversation are referred to with respect to the current utterance, e.g. the previous utterance refers to the penultimate utterance in the conversation so far;
5. NonfuzzBabe contains two pre-programmed Strings;
   - "How should I reply?"; output by NonfuzzBabe when it cannot compute a complete response to a given conversational stimulus;
   - "OK"; employed as a phatic gap filler.

4.3.2 Overview of System Operation and behaviour.

The NonfuzzBabe system comprises four modules sharing thirteen components. The modules are listed below with their hierarchy of components.

4.3.2.1 System modules and components.

- **Conversation Facilitator.**
  - Conversation Manager
  - StringCorrespondence Evaluator
  - StimulusTextCouplet Comparator
  - Variable Extractor

- **Pattern Extractor.**
  - Extractor Manager
  - DominantFeatureCouplet Pre-selector
  - DominantFeatureCouplet Detector
  - Variable Extractor

- **Pattern Comparator:**
  - ConstantEquivalence Detector
  - SpreadingActivation Generator

- **Response Hypothesiser:**
  - CompetitiveResponse Constructor
  - ConversantInteraction Manager
  - WinningResponse detector

In order to model language behaviour, NonfuzzBabe employs two distinct processing phases:

- a learning phase;
- a conversationally interactive phase.

Learning may either occur prior to a conversationally interactive phase, or instances of it may be interspersed during a conversationally interactive phase. In the latter case, the conversation must contain a section (that may be distributed throughout the conversation) that complies with the requirements of a single exposure training set as specified below. A conversation employing mixed phases is referred to as interactive learning conversation.

4.3.2.2 Learning phase.

During a learning phase, NonfuzzBabe is input with a number of example conversational utterance pairs. This corpus comprises a training set. The training set may contain any type of conversational utterance pair provided that they are represented by text comprising Characters. If the system is required to learn a language behaviour during a single exposure to one training set, then the training set must possess the following attributes:

- comprise whole conversational pairs;
- contain at least two examples of each different type of conversational pair;
- each type of conversational pair must represent a different usage of the pair.

The learning phase comprises a process of computing and storing specific textual patterns from a training corpus as follows:

- input training set to a TextCouplet database;
- compute FeatureCouplet (and possibly MetaCouplet) patterns from TextCouplets;
- output FeatureCouplet (and possibly MetaCouplet) patterns to the FeatureCouplet (and the MetaCouplet) database(s respectively).

The process may be represented by the following simplified data flow diagram (see *Figure 9: Iconic convention employed in data flow diagrams in this thesis.* on page 100):

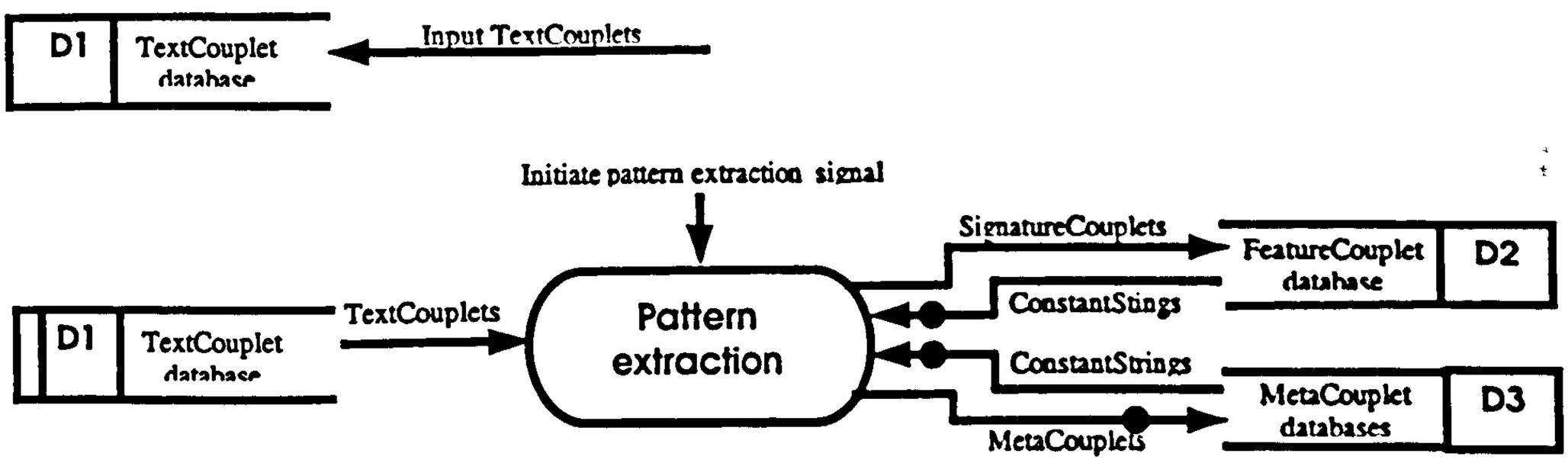


Figure 3: Simplified data flow diagram of Pattern extraction process.

A learning phase, therefore, comprises a pattern extraction process and the two terms may be considered to be synonymous in the Baby model.

4.3.2.3 Conversationally interactive phase.

NonfuzzBabe processes human current utterances in one of a number of different ways. These various processes are determined by calling particular selections of modules. The selection of modules that NonfuzzBabe utilises to process a given human utterance during a conversation, is determined by a system of utterance categorisation. The current utterance is processed according to the category of the previous utterance. There are seven categories of conversational utterance recognised by NonfuzzBabe, two relate to human produced utterances and five to NonfuzzBabe produced utterances. NonfuzzBabe automatically assigns categories to utterances. Utterance categories are specified as follows:

1. a conversational stimulus utterance made by the human;
2. a human utterance that corrects a previous utterance made by NonfuzzBabe;
3. a hypothesised response utterance made by NonfuzzBabe;
4. the "How should I reply?" pre-programmed string;
5. the "OK" pre-programmed string;
6. a rote learned response utterance made by NonfuzzBabe;
7. a NullString utterance (a virtual construct assumed to have been made by NonfuzzBabe prior to the beginning of a new conversation).

It is assumed, in this description of conversationally interactive processing, that NonfuzzBabe has previously undergone a learning phase and consequently extracted FeatureCouplets (and possibly

MetaCouplets) to their respective database(s). Each time that NonfuzzBabe processes a response to a human utterance, the human utterance must have been preceded by a category 3 or 4 or 5 or 6 or 7 utterance made by NonfuzzBabe. Categories 1 or 2 are assigned to a current human utterance according to how it interacts with, and the category of, NonfuzzBabe's previous utterance. The operation of NonfuzzBabe in conversationally interactive mode, is therefore described in terms of how current utterances are processed as determined by the category of the previous utterance.

4.3.2.3.1 Processing a current human utterance following a category 7 response.

This condition exists exclusively when the current utterance constitutes the first utterance of a conversation. Such an utterance must necessarily represent one from the human conversant and consequently NonfuzzBabe assigns them with category 1 status. Category 1 current utterances are processed as follows:

- compare every currently stored TextCouplet with every currently stored FeatureCouplet / MetaCouplet, and test whether ConstantEquivalence exists between both Strings of the TextCouplet and both SignatureParts of the FeatureCouplet / MetaCouplet;
- if ConstantEquivalence is detected then for each of its detection(s):
  store the two VariableClusters derived (along with other VariableClusters that may already exist) in the VariableParts of the respective FeatureCouplet / MetaCouplet.
- when every TextCouplet and every FeatureCouplet / MetaCouplet has been compared, if one or more occurrence(s) of ConstantEquivalence was found, then the TextCouplet(s) that shared ConstantEquivalence with SignatureParts of FeatureCouplet(s) / MetaCouplet(s) are deleted from the TextCouplet database.
- test whether the input utterance occurs identically as a StimulusTextCouplet in the TextCouplet database.
- if the input utterance occurs identically as a StimulusTextCouplet then:
  - output the ResponseTextCouplet associated with the StimulusTextCouplet that is identical to the input utterance;
  - set UtteranceType flag to category 6;
  - stop processing (awaiting a new input from the human).
- if the input utterance does not occur identically as a StimulusTextCouplet then:
  - add a copy of the current utterance to the TextCouplet database as the StimulusTextCouplet element of a potential new TextCouplet;
  - pass the current utterance to the Pattern Comparator module, that in turn provides data for the Response Hypothesiser to produce an hypothesised response.

The Pattern Comparator module and response hypothesiser module are described in more detail later in this chapter. In addition, they are described superficially here in order to provide an understanding of the system behaviour being currently described.

The **Pattern Comparator module** detects SecondaryActivatedCouplets and if they exist, stores the address(es) of their CoupletDaughter(s) in a database called the ConstantActivatedList.

The **Response Hypothesiser module** takes the ConstantActivatedList provided by the Pattern Comparator module and computes:

- a set of competitive hypothesised responses containing at least one full language structure;
- a winning hypothesised response;
- a "How should I reply?" response.

  OR
- a set of competitive hypothesised responses containing no full language structures;
- "How should I reply?" response.

Whichever set of data is output, processing stops awaiting an input from the human that instructs the Response Hypothesiser module to either:

1. accept the winning hypothesised response;
2. accept one of the competitive hypothesised responses;
3. accept the "How should I reply?" response.

Whichever response is chosen by the human, the corresponding response is added to the TextCouplet database as the ResponseTextCouplet of the StimulusTextCouplet that stimulated the output, thus forming a new complete TextCouplet.

4.3.2.3.2 Processing a current human utterance following a category 3 or 4 response.

This condition exclusively exists if the previous utterance was a response hypothesised by NonfuzzBabe. The category of the current human utterance in this condition can be either 1 or 2. NonfuzzBabe distinguishes between category 1 and 2 current utterances by employing the StringCorrespondence Evaluator that acts as a front-end component in the Conversation Facilitator module. Only the StringCorrespondence Evaluator component and associated Conversation

Facilitator Module functions are described here as other processing is identical to that described above in *processing a current human utterance following a category 7 response*:

- evaluate StringCorrespondence between NonfuzzBabe's previous utterance and the current human utterance.
- if StringCorrespondence is high, or the previous response was "How should I reply?", then the current utterance is processed as a category 2 utterance. The Conversation Facilitator handles the majority of category 2 utterance processing without calling other modules. Consequently details of the processes involved will be included here as follows:

  - the ResponseTextCouplet processed from the last human utterance, is replaced by the current human utterance in the TextCouplet database;
  - the "OK" response is output to the winning response screen interface object (see Figure7);
  - the UtteranceType flag is set to category 5;
  - if the number of TextCouplets in the TextCouplet database is equal to, or more than, a pre-set value (typically 2), then the PatternExtractor module is called;

  processing stops (awaiting a new input from the human);
- if StringCorrespondence is low and the previous response was not "How should I reply?", then the current utterance is processed as a category 1 utterance, (see *processing a current human utterance following a category 7 response* above).

4.3.2.3.3 Processing a current human utterance following a category 5 response.

This condition can only exist if the previous NonfuzzBabe utterance was the "OK" String. In this condition NonfuzzBabe assigns the current human utterance as category 1, (see *processing a current human utterance following a category 7 response* above).

4.3.2.3.4 Processing a current utterance following a category 6 response.

This condition can only exist if the previous NonfuzzBabe utterance was a rote learned response, i.e., that the ante-penultimate utterance was found as a StimulusTextCouplet in the TextCouplet database. In this condition NonfuzzBabe assigns the current human utterance as category 1 status, (see *processing a current human utterance following a category 7 response* above).

4.4 Module descriptions.

This sub-section describes the processes involved with each NonfuzzBabe module. Descriptions include a textual account and a process flow diagram. A data flow diagram of the complete system, accompanied by a data dictionary, is included at the end of the chapter. Only the function of components that are called by modules is described within this sub-section, however. For program specific levels of description, appendix A contains the appropriate code.

4.4.1 Conversation Facilitator module.

Sufficient description of the operation of the Conversation Facilitator has been given in 7.3.2. Further description of the Conversation Facilitator is limited, therefore, to an input / output specification and a description of the role of each component, followed by a process flow diagram representation:

Input: any String and a set of TextCouplets.

Output: **either:**

the input String;

UtteranceType flag set to empty;

or:

the "OK" String;

Initiate Pattern Extraction signal;

UtteranceType flag set to category 5;

or:

a corresponding ResponseTextCouplet;

UtteranceType flag set to category 6.

Component input/output mapping. For program specific levels of description, appendix A contains the appropriate code.

- the Conversation Manager organises processing routes and sequences;
- the StringCorrespondence Evaluator:

  Input: any two String.

  Output: StringCorrespondence.

- Variable Extractor:

  Input: set of String pairs and a set of FeatureCouplets.

  Output: VariableParts.

- the StimulusTextCouplet Comparator:

  Input: any String.

  Output:

  either:

  the ResponseTextCouplet of a TextCouplet whose StimulusTextCouplet is identical to input String;

  or:

  input String.

Process flow diagram

(Conversation Facilitator module)

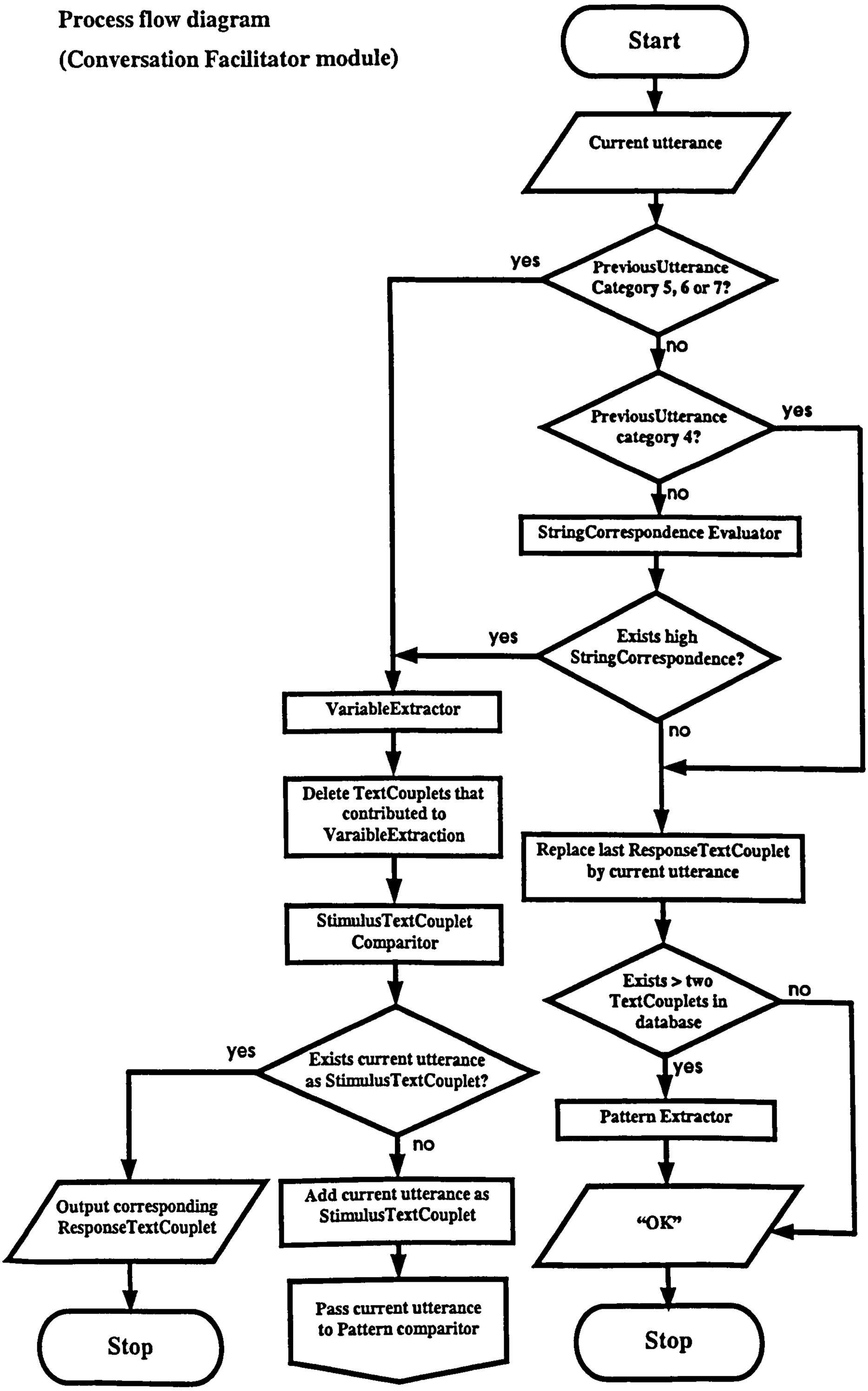


Figure 4: Process flow diagram of the Conversation Facilitator module.

4.4.2 Pattern Extractor.

The following sub-section describes the operation, specifies input / output parameters and provides a process flow diagram of the Pattern Extractor module.

Input: any set of String pairs.

Output: the set of FeatureCouplets and MetaCouplets derivable from the input set.

Component input/output mapping For program specific levels of description, appendix A contains the appropriate code.

- Extractor Manager: organises processing route sequences;

- DominantFeatureCouplet Pre-selector:

  Input: set of String pairs.

  Output: set of String pairs with largest StringCorrespondence.

- DominantFeatureCouplet Detector:

  Input: set of String pairs with largest StringCorrespondence.

  Output: DominantFeatureCouplets.

- Variable Extractor:

  Input: set of String pairs and set of FeatureCouplets.

  Output: VariableClusters

The Pattern Extractor executes the following processes in order to transform input to output data:

- copy TextCouplets stored in the TextCouplet database to the input set;
- detect from the input set, String pairs with largest StringCorrespondence and store them in a pre-select list;
- recursively apply the following process to the pre-select list until the pre-select list is empty:
  - compute DominantFeatureCouplets;
  - store DominantFeatureCouplets in FeatureCouplet / MetaCouplet databases;
  - delete String pairs that formed DominantFeatureCouplets.

This pre-select/DominantFeatureCouplets detection process is applied recursively until input set is empty. Then, if DominantFeatureCouplets were detected from the previous input set, their

ConstantParts are copied to the current input set and the pre-select/DominantFeatureCouplets detection process executes again. This process continues until no more DominantFeatureCouplets can be detected. Each time DominantFeatureCouplets are extracted from new input sets, they are progressively stored in different databases as follows:

- input TextCouplets; output FeatureCouplet database;
- input ConstantParts of FeatureCouplets; output to Meta(1)Couplet database;
- input ConstantParts of Meta(1)Couplets; output to Meta(2)Couplet database;
- input ConstantParts of Meta(2)Couplets; output to Meta(3)Couplet database; (and so on until).....;
- input ConstantParts of Meta(5)Couplets; output to Meta(6)Couplet database.

Generally, a negative correlation exists between increasing meta level and number of DominantFeatureCouplets detected at that level, consequently meta level is self-limiting. There is no model-theoretic significance to the fifth level of MetaCouplet, rather that no test corpus has so far been applied to NonfuzzBabe that produces higher than Meta(4)Couplets. It is a trivial implementation exercise to increase the level of meta detection if it should ever become necessary. When no more DominantFeatureCouplets can be detected, the next input set is empty and this signals the Extractor Manager to terminate the pattern extraction process and initiates the Variable Extractor component that executes the following algorithm:

- compare every currently stored TextCouplet with every currently stored FeatureCouplet/MetaCouplet, and test whether ConstantEquivalence exists between both Strings of the TextCouplet and both SignatureParts of the FeatureCouplet/MetaCouplet.
- if such constant equivalence exists, then for each of its occurrence(s):
  store the two VariableClusters derived in the VariableParts of the respective FeatureCouplet/MetaCouplet, (along with other VariableClusters that may already exist).
- when every TextCouplet and every FeatureCouplet/MetaCouplet has been compared, if one or more occurrence(s) of ConstantEquivalence was found, then the TextCouplet(s) that shared ConstantEquivalence with SignatureParts of FeatureCouplet(s)/MetaCouplet(s) are deleted from the TextCouplet database.

Process flow diagram.

Pattern Extractor module

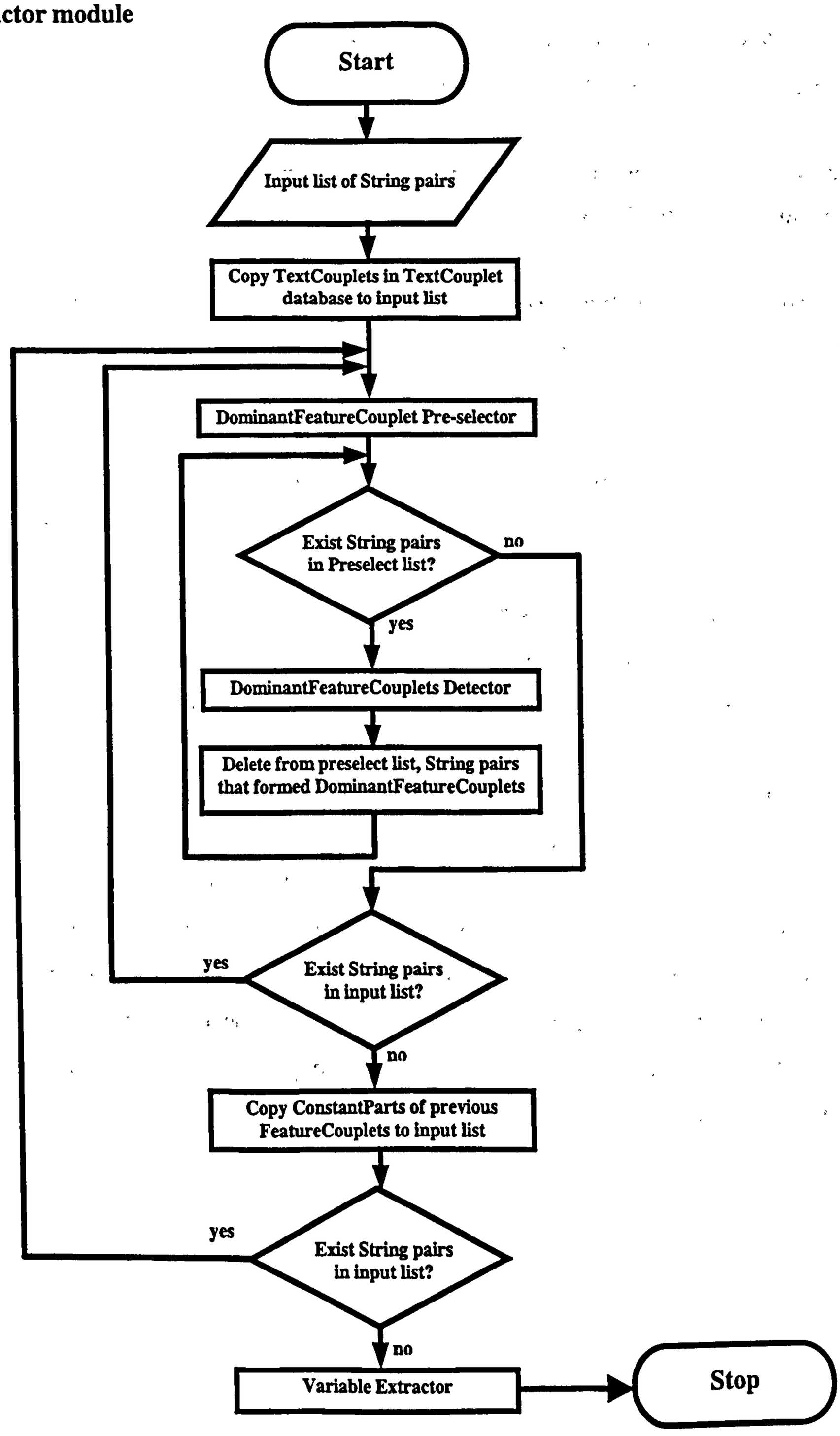


Figure 5: Process flow diagram of Pattern Extractor module.

4.4.3 Pattern Comparator.

The following sub-section describes the operation, specifies input / output parameters and provides a process flow diagram of the Pattern Comparator module.

Input: any String and a set of FeatureCouplets/MetaCouplets.
Output: ConstantEquivalenceList.

Component input/output mapping: For program specific levels of description, appendix A contains the appropriate code.

- ConstantEquivalence Detector:
  Input: a String and a set of FeatureCouplets/MetaCouplets.
  Output: ConstantEquivalenceList containing PrimaryActivatedCouplets.

- SpreadingActivation Generator:
  Input: a set of Features and PrimaryActivatedCouplets.
  Output: ConstantEquivalenceList containing SecondaryActivatedCouplets.

The ConstantEquivalence Detector component tests for ConstantEquivalence between input and the SignaturePart of any FeatureCouplet (and possibly any MetaCouplet) stored in the FeatureCouplet (and MetaCouplet) database(s). Such couplets are labelled PrimaryActivatedCouplets. The address of the CoupletDaughter of PrimaryActivatedCouplets in the FeatureCouplet/MetaCouplet database, is stored in a database called the ConstantEquivalenceList. The SpreadingActivation Generator components detects SecondaryActivatedCouplets. If such occurrences are found, the address(s) of each of their CoupletDaughters in the FeatureCouplet/MetaCouplet database is added to the ConstantEquivalenceList.

Process flow diagram.

(Pattern Comparator module)

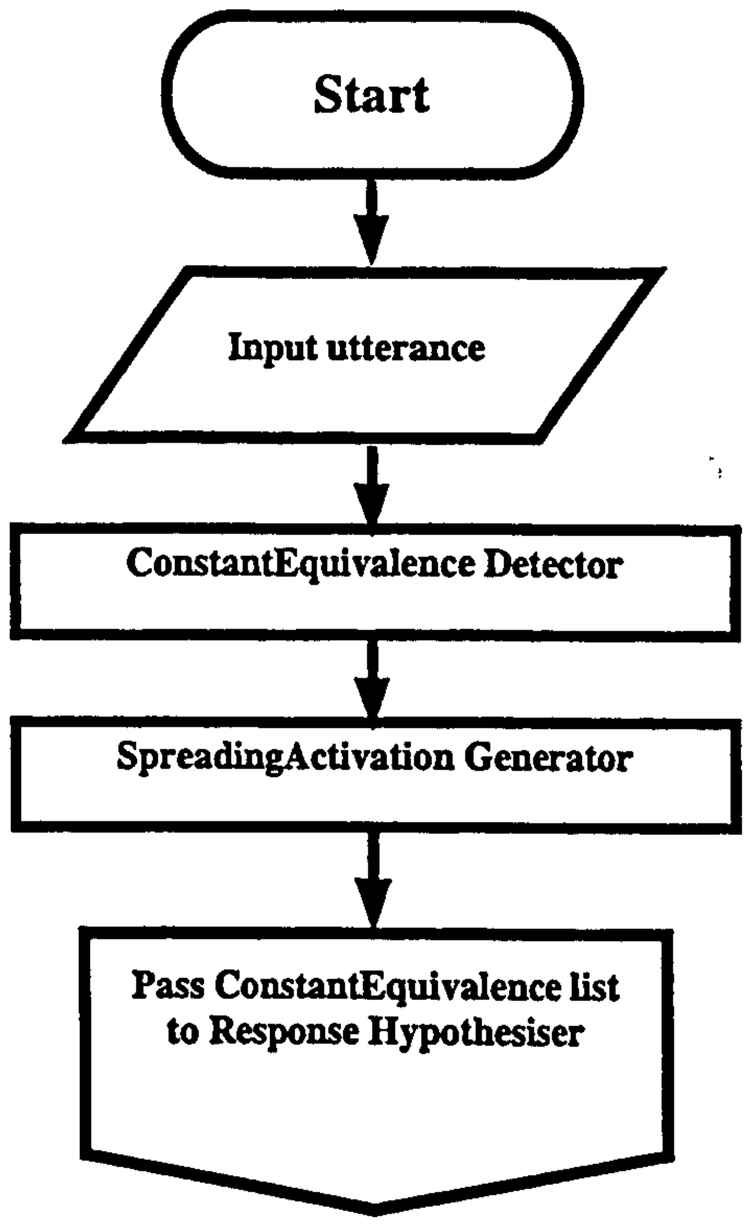


Figure 6: Process flow diagram of Pattern Comparator module.

4.4.4 Response Hypothesiser

The following sub-section describes the operation, specifies input / output parameters and provides a process flow diagram of the Response Hypothesiser module. In addition, because the description of this module includes mention of screen interface objects for the first time (in exception of one case), the NonfuzzBabe interface is shown and labelled with such interface objects as follows:

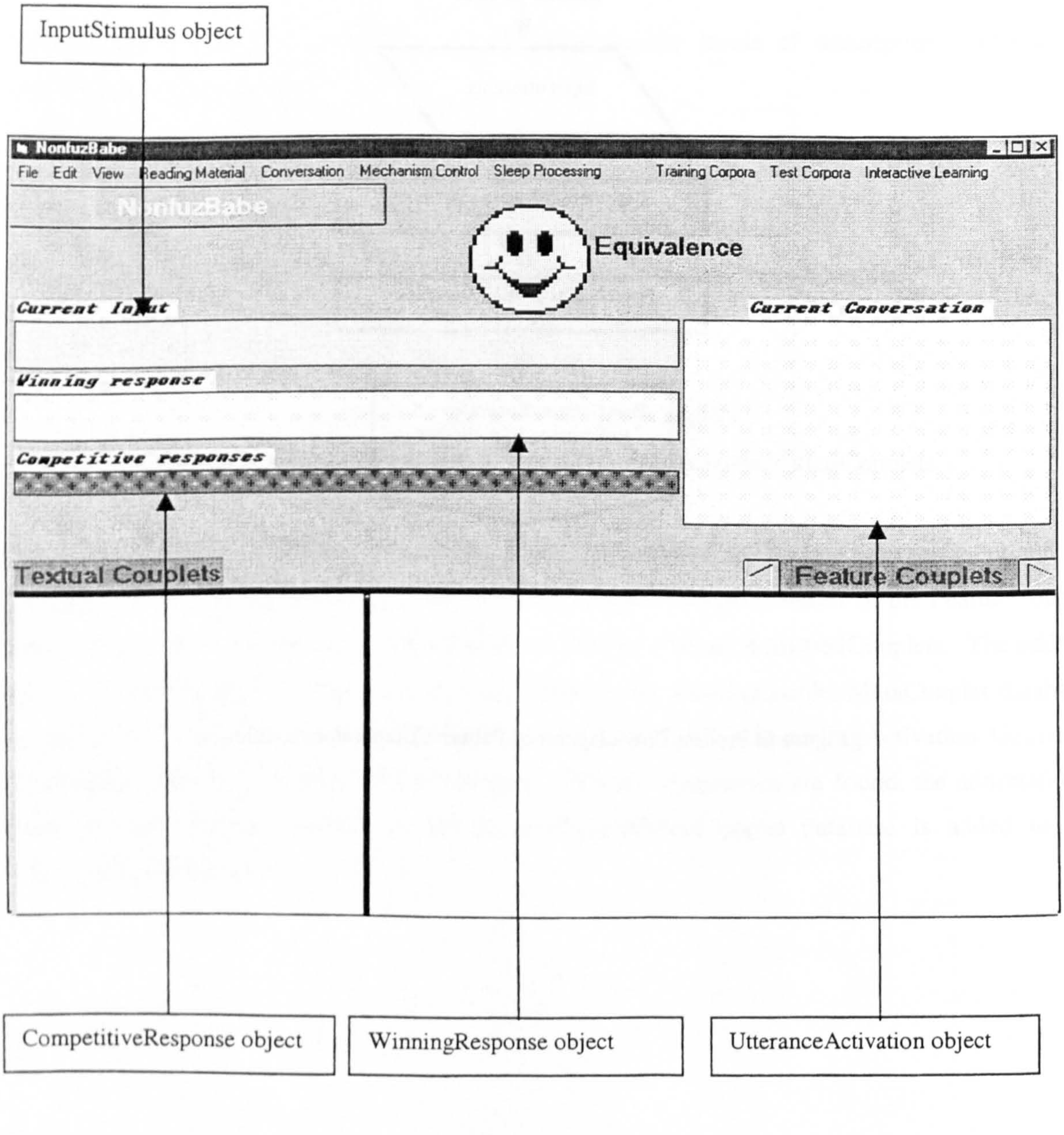


Figure 7: NonfuzzBabe screen display.

Input:

- ConstantEquivalenceList;
- a set of FeatureCouplets/MetaCouplets;
- human mediated hypothesised response.

Output: **either:**

- human mediated hypothesised response;
- UtteranceType flag set to category 3.

or:

- the "How should I reply?" String;
- UtteranceType flag set to category 4.

Component input/output mapping: For program specific levels of description, appendix A contains the appropriate code.

- CompetitiveResponse constructor:

  Input: ConstantEquivalenceList and a set of FeatureCouplets/MetaCouplets.

  Output: CompetitiveResponseList containing a set of hypothesised responses.

- WinningResponse detector:

  Input: the CompetitiveResponseList

  Output: WinningResponse.

- ConversantInteraction manager:

  Input: WinningResponse and human mediation.

  Output: human mediated hypothesised response.

The Response Hypothesiser module takes the ConstantEquivalenceList provided by the Pattern Comparator module and a set of FeatureCouplets/MetaCouplets, and derives all computable language structures (including part structures) and outputs them to the CompetitiveResponse screen interface object. A list is computed comprising each different full language structure ordered by its frequency of occurrence in the CompetitiveResponse screen interface object. This ordered list is displayed in the UtteranceActivation screen interface object. The most frequently occurring, complete language structure is output to the WinningResponse screen interface object as the hypothesised response. If two or more different complete language structures exist having the same frequency of occurrence in the CompetitiveResponse screen interface object, then a first-in-first-out principle is employed to compute the winning response. If no full language structure has been

derived, then although part structures are displayed in the CompetitiveResponse screen interface object, the Response Hypothesiser returns the "How should I reply?" String to the WinningResponse screen interface object. At this stage of processing, the Response Hypothesiser makes available the following data:

- a set of competitive hypothesised responses containing at least one full language structure;
- a winning hypothesised response.

OR

- a set of competitive hypothesised responses containing no full language structures;
- a "How should I reply?" response.

Whichever set of data is submitted by the Response Hypothesiser, processing is suspended awaiting an input from the human that instructs the Response Hypothesiser to either:

1. accept the winning hypothesised response;
2. accept one of the competitive hypothesised responses;
3. accept the "How should I reply?" response.

If the human chooses option 1 or 2, then the Response Hypothesiser allocates category 3 to the UtteranceType flag else it allocates category 4 to the UtteranceType flag. Whichever response is chosen by the human, it is added to the TextCouplet database as the ResponseTextCouplet of the StimulusTextCouplet that stimulated the output, thus forming a complete TextCouplet.

Process flow diagram.

Response Hypothesiser

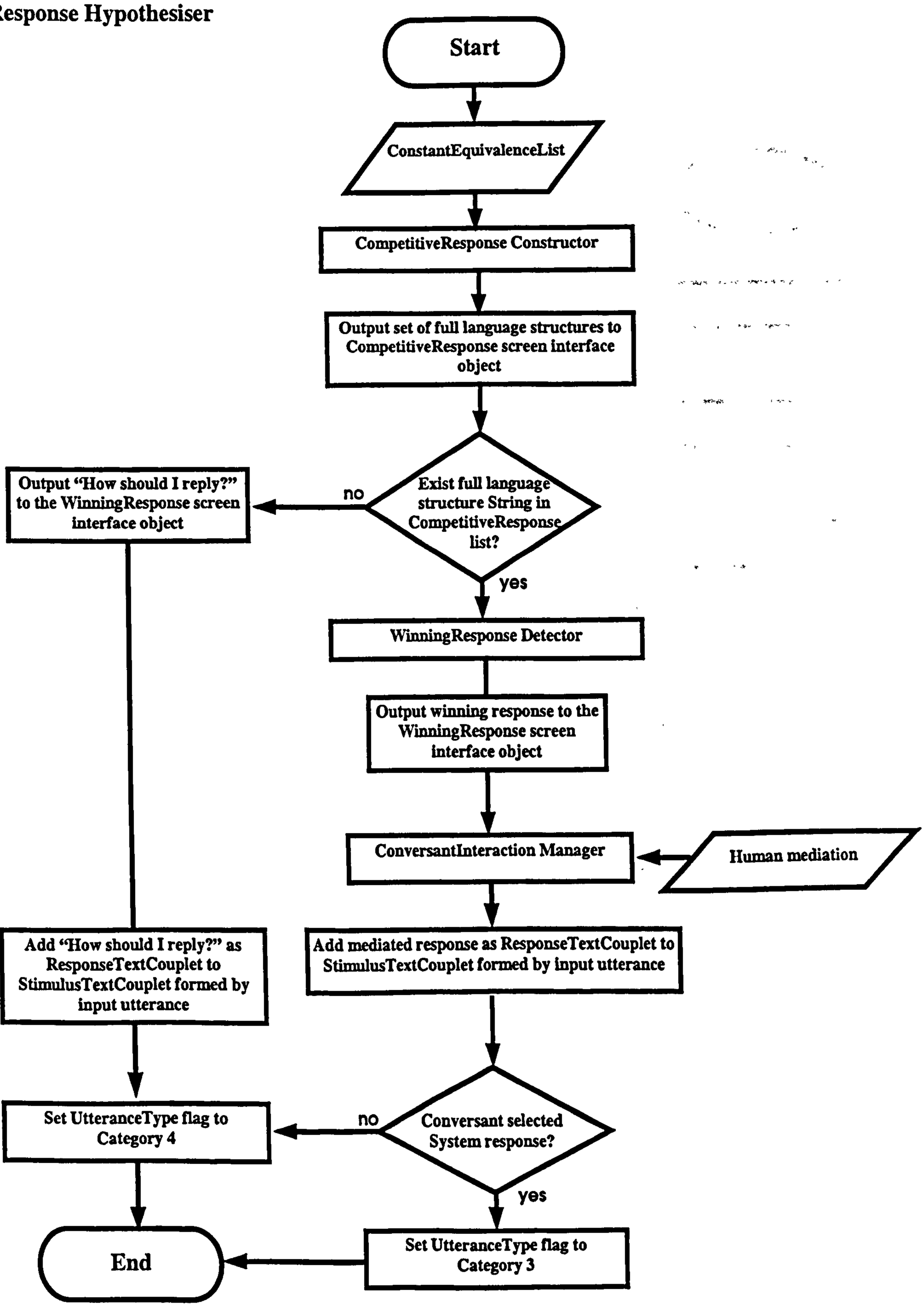


Figure 8: Flow diagram of Response Hypothesiser module.

4.5 System Data flow diagram.

There follows a diagram that shows the data flow within the complete NonfuzzBabe system. Figure 9 shows the iconic convention employed (Sommerville, 1989), is as follows:

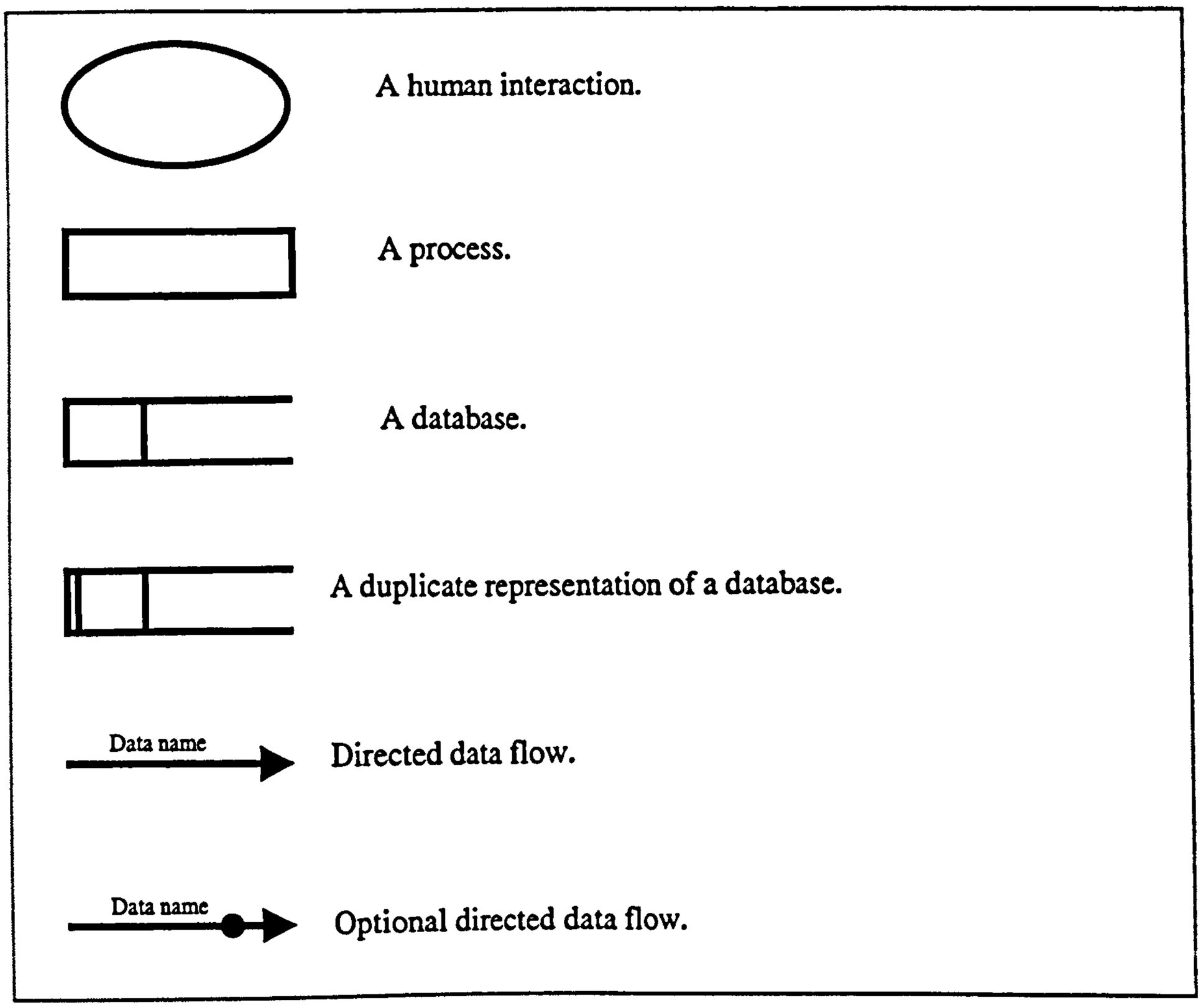
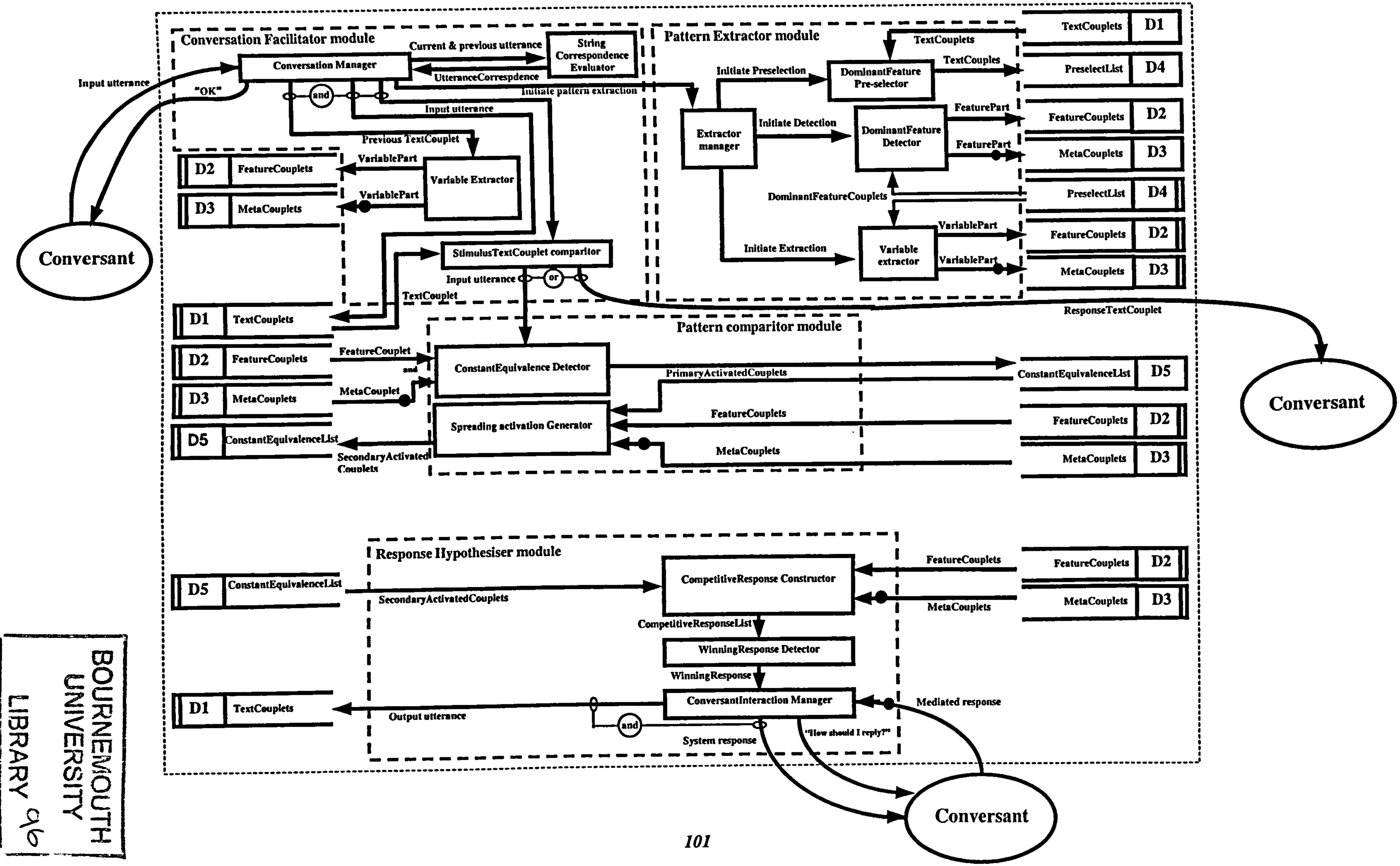


Figure 9: Iconic convention employed in data flow diagrams in this thesis.

Figure 10: NonfuzzBabe Data Flow Diagram.



4.6 System Data Dictionary.

No.	DFD Name (entity name)	Name in VB	Data Type	Variable Type	Data description
D1	TextCouplets	lstdb(0)	String	List	Input utterances that either: a) have not been input to the Pattern Extraction process; b) share insufficient commonality to form FeatureCouplets.
D2	FeatureCouplets	lstdb(1)	String	List	Output of the Pattern Extraction process, input with TextCouplets.
D3	MetaCouplets	lstdb(2) - lstdb(7)	String	List	Output of the Pattern Extraction process, input with FeatureCouplets or MetaCouplets
D4	Pre-selectList	Buffer 1	String	List	DominantFeatureCouplets
D5	ConstantEquivalents	List8(0) - List8(2)	Single	List Array	list8(0) - Base location of SecondaryActivatedCouplets. list8(1) - List location of SecondaryActivatedCouplets. list8(2) - 1

Table 3: Data dictionary for NonfuzzBabe.

4.7 Summary.

In this chapter, the NonfuzzBabe system has been described in some detail. Firstly, the rationale that underpins the implementation were reviewed, then terms used with context order sensitive meanings were defined. An overview of the system behaviour was then given followed by a separate specification for each of the modules that comprise the system. Finally a data flow diagram and data dictionary were presented. For program specific levels of description, appendix A contains the appropriate code.

Chapter 4 - NonfuzzBabe

5.0 Results of NonfuzzBabe's language performance.

This chapter shows results that were obtained from experiments conducted upon NonfuzzBabe in order to test the first hypothesis stated in chapter *3.0 Research strategy.*, i.e. that:

> Baby models, implemented with equivalence classification techniques will, following a pattern extraction process on a representative corpus, be capable of hypothesising correct responses to simple, novel textual representations of conversationally interactive stimuli, as judged by a human user of the language employed.

The chapter includes a specification of the tests employed, the results obtained and an analysis of the results. Finally, a summary is given. The results are not discussed in this chapter, as this occurs later in chapter *7.0 Language performance of NonfuzzBabe, CorrBabe and FuzzBabe.*

5.1 Specification of tests.

In order to model the simple nature of interaction required by the experimental hypothesis, it was decided to employ language that exampled basic syntactic transformation. The types of syntactic transformation were selected randomly. Also it should be noted that examples of each type of syntactic transformation were contrived to minimise the occurrence of random Constant elements within Variables. For example, it will be noted that all the verbs used in test 1 begin with a different character. This action was required because of the small size of the training set used. Whilst this is satisfactory for testing the experimental hypothesis, it raises important issues concerning model scaling. These issues are resolved when NonfuzzBabe is applied to random selections of much larger corpora in section *7.2 Results of comparative tests using larger training and test corpora.* The syntactic transformations chosen for the test were:

- subject verb agreement formation;
- past tense formation;
- plural formation;
- simple translation;
- passive formation.

In addition to testing NonfuzzBabe's language performance with these simple syntactic transformations, two further aspects of NonfuzzBabe's language behaviour was also tested. This was achieved by applying plural and passive syntactic transformations in more complex ways, i.e.:

- **sensitivity:** the ability to process minor cluster transformations within a main class of syntactic transformation, e.g. **deny → denies and flay → flays;**
- **selectivity:** the ability to process more that one syntactic transformation following a single learning exposure to these syntactic transformations.

Each test comprised three parts:

1. input a training corpus:
   where each training corpus complies with the requirements of a single exposure training set as defined in *4.3.2.2 Learning phase.*, i.e. that they comprise whole conversational pairs; contain at least two examples of each different type of conversational pair; each type of conversational pair represents a different usage of the pair;

2. initiate the pattern extraction process;

3. test with conversational stimuli that include examples contained in the training set and examples similar to, but disjoint from, the training set:

5.2 Test results.

There follows a series of tables showing detail of the tests applied to NonfuzzBabe along with the results obtained.

5.2.1 Test 1: Subject-verb agreement formation:

No.	Human utterance	NonfuzzBabe's reply
1a	What is the third person singular of sing?	
1b	The third person singular of sing is sings.	
2a	What is the third person singular of talk?	
2b	The third person singular of talk is talks.	
3a	What is the third person singular of touch?	
3b	The third person singular of touch is touches.	
4a	What is the third person singular of lunch?	
4b	The third person singular of lunch is lunches.	
5a	What is the third person singular of went?	
5b	The third person singular of went is went.	
	Pattern Extraction Initiated	
6	What is the third person singular of sing?	The third person singular of sing is sings.
7	What is the third person singular of hear?	The third person singular of hear is hears.
8	What is the third person singular of touch?	The third person singular of touch is touchs.
9	What is the third person singular of fetch?	The third person singular of fetch is fetchs.
10	What is the third person singular of went?	The third person singular of went is went.

Table 4: Subject-verb agreement formation test.

5.2.2 Test 2: Past tense formation:

No.	Human utterance	NonfuzzBabe's reply
1a	What is the past tense of jump?	
1b	The past tense of jump is jumped.	
2a	What is the past tense of bank?	
2b	The past tense of bank is banked.	
3a	What is the past tense of trace?	
3b	The past tense of trace is traced.	
4a	What is the past tense of refuse?	
4b	The past tense of refuse is refused.	
5a	What is the past tense of mow?	
5b	The past tense of mow is mown.	
6a	What is the past tense of know?	
6b	The past tense of know is known.	
7a	What is the past tense of go?	
7b	The past tense of go is gone.	
	Pattern Extraction Initiated	
8	What is the past tense of jump?	The past tense of jump is jumped.
9	What is the past tense of launch?	The past tense of launch is launched.
10	What is the past tense of trace?	The past tense of trace is traceed.
11	What is the past tense of stone?	The past tense of stone is stoneed.
12	What is the past tense of mow?	The past tense of mow is mowed.
13	What is the past tense of sow?	The past tense of sow is sowed.
14	What is the past tense of go?	The past tense of go is gone.

Table 5: Past tense formation test.

5.2.3 Test 3: Plural formation:

No.	Human utterance	NonfuzzBabe's reply
1a	What is the plural of snake?	
1b	The plural of snake is snakes.	
2a	What is the plural of cow?	
2b	The plural of cow is cows.	
3a	What is the plural of datum?	
3b	The plural of datum is data.	
4a	What is the plural of agendum?	
4b	The plural of agendum is agenda.	
5a	What is the plural of deer?	
5b	The plural of deer is deer.	

Pattern Extraction Initiated

6	What is the plural of snake?	The plural of snake is snakes.
7	What is the plural of net?	The plural of net is nets.
8	What is the plural of agendum?	The plural of agendum is agenda.
9	What is the plural of septum?	The plural of septum is septa.
10	What is the plural of deer?	The plural of deer is deers.

Table 6: Plural formation test.

5.2.4 Test 4: Bi-lingual transformation:

No.	Human utterance	NonfuzzBabe's
1a	The French for speak is parler, what is the French for we speak?	
1b	Nous parlons.	
2a	The French for reside is resider, what is the French for we reside?	
2b	Nous residons.	
3a	The French for grow is grandir, what is the French for we grow?	
3b	Nous grandissons.	
4a	The French for finish is finir, what is the French for we finish?	
4b	Nous finissons.	
5a	The French for sleep is dormir, what is the French for we sleep?	
5b	Nous dormons.	

Pattern Extraction Initiated

6	The French for speak is parler, what is the French for we speak?	Nous speakons.
7	The French for give is donner, what is the French for we give?	Nous giveons.
8	The French for grow is grandir, what is the French for we grow?	Nous growons.
9	The French for fatten is grossir, what is the French for we fatten?	Nous fattenons.
10	The French for sleep is dormir, what is the French for we sleep?	Nous sleepons.

Table 7: Bi-lingual transformation test.

5.2.5 Test 5: Passive transformation.

No.	Human utterance	NonfuzzBabe's reply
1a	John kicks the ball.	
1b	The ball is kicked by John.	
2a	Henry hails the cab.	
2b	The cab is hailed by Henry.	
3a	Pat sprays the cottage.	
3b	The cottage is sprayed by Pat.	
4a	Steve lands the boat.	
4b	The boat is landed by Steve.	
5a	Rob fetches the gun.	
5b	The gun is fetched by Rob.	
6a	Marty types the message.	
6b	The message is typed by Marty.	
7a	Angela rates the music.	
7b	The music is rated by Angela.	
8a	Quentin dares the fellow.	
8b	The fellow is dared by Quentin.	
	Pattern Extraction Initiated	
9	John kicks the ball.	The ball is kicked by John.
10	April locks the door.	The door is locked by April.
11	Steven hooks the fish.	The fish is hooked by Steven.
12	Quentin dares the fellow.	The fellow is dared by Quentin.
13	Jim lines the pavement.	The pavement is lined by Jim.
14	Carol flames the steak.	The steak is flamed by Carol.
15	David runs the roost.	The roost is runed by David.

Table 8: Passive formation test.

5.2.6 Test 6: Sensitivity:

No.	Human utterance	NonfuzzBabe's reply
1a	What is the plural of bus?	
1b	The plural of bus is buses.	
2a	What is the plural of fly?	
2b	The plural of fly is flies.	
3a	What is the plural of play?	
3b	The plural of play is plays.	
4a	What is the plural of stress?	
4b	The plural of stress is stresses.	
5a	What is the plural of goat?	
5b	The plural of goat is goats.	
6a	What is the plural of dry?	
6b	The plural of dry is dries.	
7a	What is the plural of nay?	
7b	The plural of nay is nays.	
8a	What is the plural of joy?	
8b	The plural of joy is joys.	
9a	What is the plural of etch?	
9b	The plural of etch is etches.	
10a	What is the plural of cow?	
10b	The plural of cow is cows.	
11a	What is the plural of toy?	
11b	The plural of toy is toys.	
12a	What is the plural of hunch?	
12b	The plural of hunch is hunches.	

Pattern Extraction Initiated

13	What is the plural of truss?	The plural of truss is trusses.
14	What is the plural of deny?	The plural of deny is denys.
15	What is the plural of flay?	The plural of flay is flays.
16	What is the plural of breed?	The plural of breed is breeds.
17	What is the plural of saveloy?	The plural of saveloy is saveloys.
18	What is the plural of bench?	The plural of bench is benches.

Table 9: Sensitivity test.

5.2.7 Test 7: Selectivity.

No.	Human utterance	NonfuzzBabe's reply
1a	John kicks the ball.	
1b	The ball is kicked by John.	
2a	What is the past tense of jump?	
2b	The past tense of jump is jumped.	
3a	Quentin dares the fellow.	
3b	The fellow is dared by Quentin.	
4a	What is the past tense of bank?	
4b	The past tense of bank is banked.	
5a	Angela rates the music.	
5b	The music is rated by Angela.	
6a	Henry hails the cab.	
6b	The cab is hailed by Henry.	
7a	What is the past tense of mow?	
7b	The past tense of mow is mown.	
8a	Steve lands the boat.	
8b	The boat is landed by Steve.	
9a	Marty types the message.	
9b	The message is typed by Marty.	
10a	What is the past tense of know?	
10b	The past tense of know is known.	
11a	Pat sprays the cottage.	
11b	The cottage is sprayed by Pat.	
12a	Rob fetches the gun.	
12b	The gun is fetched by Rob.	
13a	David runs the roost.	
13b	The roost is run by David.	
	Pattern Extraction Initiated	
14	What is the past tense of jump?	The past tense of jump is jumped.
15	What is the past tense of breach?	The past tense of breach is breached.
16	Henry hails the cab.	The cab is hailed by Henry.
17	Peter saves the day.	The day is saved by Peter.
18	What is the past tense of sow?	The past tense of sow is sown.
19	Mary types the memo.	The memo is typed by Mary

Table 10: Selectivity test.

5.3 Analysis of results.

The results obtained from applying these seven tests is be summarised in Table 11. This table is employed again in section *7.1 Results of comparative tests with NonfuzzBabe.*, for comparing NonfuzzBabe's basic language performance with that of CorrBabe and FuzzBabe.

Syntactic transformation test	Main comments	Accuracy
Subject verb agreement	One of two transformation rules learnt; Exception learnt.	60%
Past tense formation	One of three transformation rules learnt; Exception learnt.	43%
Plural formation	Two of two transformation rules learnt; Exception not learnt.	80%
Bi-lingual transformation	No transformation rules learnt; Mixed language responses.	0%
Passive transformation	Two of two transformation rules learnt; Exception not learnt.	86%
Sensitivity	Five of six transformation rules learnt No exception tested	83%
Selectivity	Four of four transformation rules learnt; No exception tested	100%

Table 11: Accuracy performance of NonfuzzBabe when applied to simple syntactic transformation tests.

5.4 Summary.

The requirements set out in section *3.1 Research domain.* for tests performed upon NonfuzzBabe were to explore the following:

- whether NonfuzzBabe could process language at any level, and if so;
- indicate a reference language performance, against which Baby implementations using fuzzy classification techniques could be compared.

It is argued that the results shown satisfy each of the issues raised, i.e. that:

- Baby is evidently processing at some level of competence, in as much as its output cannot be seen to emulate a human non-language user;
- Table 11 indicates a language performance;
- such tests can be reapplied to progressively more fuzzy classification methods within the Baby model for the purpose of comparison.

6.0 Baby implementations employing fuzzy classification techniques.

The purpose of this chapter is to describe the CorrBabe and FuzzBabe systems. The description specifies the operation of only those modules that differ between these systems and NonfuzzBabe, as described in chapter *4.0 NonfuzzBabe.* The modules that vary, implement degrees of fuzziness in the classification algorithm (as set out in chapter *3.0 Research strategy.*), plus fuzzy inference and defuzzification components.

NB. The convention adopted in the remainder of this chapter is for the term *Correspondence* to be used in relation to CorrBabe classification and the term *FuzzyCorrespondence* to be used in relation to FuzzBabe classification.

The format of the chapter is as follows:

- review of systems rationale;
- definitions of new terms used;
- a brief overview of fuzzy processing;
- fuzzy techniques employed in Baby;
- implementation of fuzzy techniques in Baby;
- for each of the modules that differ with NonfuzzBabe the following is given:
  - a specification;
  - a process flow representation;
  - component input/output specification;
- a system data flow diagram for both CorrBabe and FuzzBabe;
- a system data dictionary for both CorrBabe and FuzzBabe.

The chapter concludes with a summary of the main points covered.

6.1 Rationale.

CorrBabe and FuzzBabe were designed, implemented and tested in order to gather data that could be used to test the second hypothesis relating to the research domain of this project, i.e. that:

> Baby models, implemented with fuzzy classification techniques, will produce superior simple language processing performance compared with Baby models implemented using equivalence classification techniques, as judged by a human user of the language employed.

It was hypothesised in section *2.10.7 Analysis of formalisms capable of processing nonmonotonic problem spaces.*, that although fuzzy classification techniques may better model the concept represented by the words "most closely correspond" in Baby's underpinning theory, possibility theory formalisms, such as fuzzy approaches, may suffer from inaccuracy and/or overproductivity. In order to test for trends in these counter effects, CorrBabe and FuzzBabe were designed to implement increasing degrees of fuzziness in their classification algorithm. A specification of the membership function for CorrBabe's and FuzzBabe's classification set is repeated here for convenience:

Given:

Two sets A and B on a universe of discourse of Strings Z;

Two sets D and E on a universe of discourse of Features Y;

Where:

$\beta \in B$ (a singleton);

D = The set of Features derived from A;

$\varepsilon \in E$ = members of D that correspond with β;

$A \cap B$ = the null set ϕ;

$E \subseteq D$.

Then:

The membership function of set E [$\mu_E(\varepsilon)$] in CorrBabe may be stated:

A Feature $\varepsilon \in E$ iff ***a concatenated element that shares the two left-most characters of EACH*** constant in the SignaturePart of the Feature ε exist in the same order, left to right, in β.

and:

The membership function of set E [$\mu_E(\varepsilon)$] in FuzzBabe may be stated:

A Feature $\varepsilon \in E$ iff ***a concatenated element that shares the two left-most characters of ANY*** constant in the SignaturePart of the Feature ε exist in the same order, left to right, in β.

6.2 Definitions.

In addition to the definitions set out in chapter *4.0 NonfuzzBabe.*, there follows a list of definitions for terms that describe concepts that are invoked for CorrBabe and FuzzBabe as follows:

1. **LocalCorrespondence ratio:**

   Given any **String** ***A$*** and a **SignaturePart** of a **Feature** ***B$*** then:

   In the case where a concatenated element that shares the two left-most characters of each constant of ***B$*** exists in the same order, left to right, in ***A$*** then the **LocalCorrespondence ratio** is defined as:

$$\frac{\text{the length of common sub-String between } A\$ \text{ and } B\$}{\text{the number of characters in } B\$}$$

   e.g. Where is your hat? or **Where's your hat**

   [Where is your]1[at?] **[Where is your]1[at?]**

   returns (17/17) = 1.0 **returns (7/17) = 0.4118**

2. **GlobalCorrespondence ratio:**

   Given any **String** ***A$*** and a **SignaturePart** of a **Feature** ***B$*** that is a member of a set (*X*) of **Features** whose **SignaturePart** possess a **LocalCorrespondence ratio** with ***A$***, and the member of *X* that shares the largest **LocalCorrespondence ratio** with ***A$*** is ***C$***, then the **GlobalCorrespondence ratio** between ***A$*** and ***B$*** is defined as:

$$\frac{\text{the length of common sub-String between } A\$ \text{ and } B\$}{\text{the length of } C\$}$$

   e.g. for the set of **SignatureParts:**

   [What is the plural of][y?]

   [What is the plural of][oy?],

   The **GlobalCorrespondence ratio** obtained with the **String**

   "What is the plural of saveloy?"

   is as follows:

   What is the plural of saveloy? and **What is the plural of saveloy?**

   [What is the plural of][y?] **[What is the plural of][oy?]**

   returns = (24/25) = 0.96 **returns = (25/25) = 1.0**

3. **LocalFuzzyCorrespondence ratio:**

   Given any **String** ***A$*** and a **SignaturePart** of a **Feature** ***B$*** then:

   In the case where a concatenated element that shares the two left-most characters of at least one constant of ***B$*** exists in the same order, left to right, in ***A$*** then the **LocalFuzzyCorrespondence ratio** is defined as:

$$\frac{\text{the length of common sub-Strings between } A\$ \text{ and } B\$}{\text{the length of } B\$}$$

e.g. Where's your hat	or	**Where's your comic**
[Where is your]1[at?]		**[Where is your]1[at?]**
returns (7/17) = 0.4118		**returns (5/17) = 0.2941**

4. **GlobalFuzzyCorrespondence ratio:**

Given any **String** ***A$*** and a **SignaturePart** of a **Feature** ***B$*** that is a member of a set (*X*) of **Features** whose **SignaturePart** possess a **LocalFuzzyCorrespondence ratio** with ***A$***, and the member of *X* that shares the largest **LocalFuzzyCorrespondence ratio** with ***A$*** is ***C$***, then the **GlobalFuzzyCorrespondence ratio** between ***A$*** and ***B$*** is defined as:

$$\frac{\text{the length of common sub-String between } A\$ \text{ and } B\$}{\text{the length of } C\$}$$

e.g. for the set of **SignatureParts:**

[What is the plural of][?]

[What is the plural of][oy?],

The **GlobalFuzzyCorrespondence** ratio obtained with the **String**

"What's the plural of saveloy?"

is as follows:

What's the plural of saveloy?	and	**What's the plural of saveloy?**
[What is the plural of][?]		**[What is the plural of][oy?]**
returns = (4/25) = 0.16		**returns = (7/25) = 0.28**

5. **Correspondence ratio.**

Given any **String** ***A$*** and a **SignaturePart** of a **Feature** ***B$***, such that their **LocalCorrespondence ratio** = C and their **GlobalCorrespondence ratio** = D, then the **Correspondence ratio** between ***A$*** and ***B$*** is defined as:

$$C \bullet D$$

6. **FuzzyCorrespondence ratio.**

Given any **String** ***A$*** and a **SignaturePart** of a **Feature** ***B$***, such that their **LocalFuzzyCorrespondence ratio** = E and their **GlobalFuzzyCorrespondence ratio** = F, then the **FuzzyCorrespondence ratio** between ***A$*** and ***B$*** is defined as:

$$E \bullet F$$

6.3 Fuzzy processing in Baby.

The Pattern Extraction process in CorrBabe and FuzzBabe is identical to NonfuzzBabe. Fuzziness is implemented in the Baby model during Conversationally Interactive processing.

6.3.1 Fuzzification.

Fuzzification is performed by producing two sets of fuzzy values that indicate the likelihood of an input utterance corresponding to a FeatureCouplet(s) (or possibly MetaCouplet(s)) in the FeatureCouplet and MetaCouplet databases. In the case of CorrBabe, the Correspondence detector component is responsible for evaluating the GlobalCorrespondence ratio and the LocalCorrespondence for each Feature (and possibly MetaFeature). In the case of FuzzBabe, the FuzzyCorrespondence detector component is responsible for evaluating the GlobalFuzzyCorrespondence ratio and the LocalFuzzyCorrespondence ratio for each Feature (and possibly MetaFeature).

6.4.1.1 Local and global correspondence metrics.

In the case of equivalence classification, the membership function of the classified set is crisp. One of the consequences of this, is that a *COTLPin* based classification evaluation is relatively uncomplicated. It is simply required to identify the Feature(s) that possess an associated SignaturePart that share ConstantEquivalence with the current input utterance, and add the address(es) of those Feature(s) to the ConstantEquivalenceList. In the case of fuzzy classification, however, the membership function of the classified set is, by definition, fuzzy. Under this condition, a *COTLPin* based classification evaluation is more complex.

COTLPin prescribes that Baby attends to the largest *subset* of stimuli within a *supra set* of competing stimuli. Complexity arises from the process of identifying the largest *subset* of corresponding Features. For example, consider a given input utterance being processed with a set of FeatureCouplets. There may be, for instance, ConstantEquivalence between the utterance and a SignaturePart of a one of the Features, giving a local correspondence fuzzy value of 1. However, there may be a larger set of Features that exhibit a correspondence metric that is high (but less than 1), such that in their totality, they represent a larger subset of correspondence than the smaller set of Features whose SignaturePart with a higher correspondence metric.

In order to properly model *COTLPin*, it is necessary, therefore, to evaluate a GlobalCorrespondence metric for each Feature, i.e. to:

- evaluate the LocalCorrespondence between the input utterance and each Feature;
- record the largest LocalCorrespondence

  and use this metric to:
- evaluate the GlobalCorrespondence between the input utterance and each Feature.

6.3.2 Fuzzy inference.

Baby may be considered as a fuzzy system with one hand crafted rule as follows:

If LocalCorrespondence **AND** GlobalCorrespondence **THEN** the ConstantCorrespondence.

So as to model the AND connective in CorrBabe and FuzzBabe, the max-product composition rule has been applied to the local and global correspondence metrics to provide a fuzzy value that models *COTLPin* for each Feature for a given input utterance.

6.3.3 Defuzzification.

In CorrBabe and FuzzBabe, the max-membership principle has been applied to the local and global correspondence metrics to provide a crisp value that models *COTLPin* for each Feature for a given input utterance. The main reasons for this are that the method is suitable for application to peak membership functions and require relatively small computational expense (Hellendoorn & Thomas, 1993).

6.4 Fuzzy implementations of Baby.

NonfuzzBabe was implemented in such a way as to enable modification to fuzzy classification relatively uncomplicated, only the Pattern Comparator module requires attention. It will be recalled that the Pattern Comparator module of NonfuzzBabe contained two components:

- ConstantEquivalence Detector;
- SpreadingActivation Generator.

In CorrBabe, the Pattern Comparator module comprises:

- Correspondence Detector;
- FuzzyInference component;
- Defuzzification component;
- SpreadingActivation Generator.

In FuzzBabe, the Pattern Comparator module comprises:

- FuzzyCorrespondence Detector;
- FuzzyInference component;
- Defuzzification component;
- SpreadingActivation Generator.

The difference between the Pattern Comparator as implemented in CorrBabe and FuzzBabe is therefore restricted to the components Correspondence Detector and FuzzyCorrespondence Detector. Given an input utterance and set of FeatureCouplets, the Correspondence Detector produces a database called the CorrespondenceList or FuzzyCorrespondenceList that contains Feature address(es) with their LocalCorrespondence and GlobalCorrespondence ratios or LocalFuzzyCorrespondence and GlobalFuzzyCorrespondence ratios respectively.

It can be seen that these modifications entail two new modules:

- FuzzyInference component;
- Defuzzification component.

Both these modules were implemented identically in CorrBabe and FuzzBabe. The following sections detail the operation of the Pattern Comparator module as implemented in CorrBabe and FuzzBabe and the FuzzyInference and Defuzzification components. Only the function of components that are called by these modules are described. For program specific levels of description, appendix A contains the appropriate code. All other aspects of CorrBabe and FuzzBabe are identical with NonfuzzBabe and will not be discussed further here.

6.4.1 Modification of the Pattern Comparator module.

The following sub-section describes the operation, specifies input / output parameters and provides a process flow diagram of the Pattern Comparator module for both CorrBabe and FuzzBabe.

Input: any String and a set of FeatureCouplets.

Output: SecondaryCrispList.

Figure 11 and Figure 12 show a process flow diagram for the Pattern Comparator module implemented in CorrBabe and the Pattern Comparator module implemented in FuzzBabe respectively.

6.4.1.1 The Correspondence detector as implemented in CorrBabe:

Input: any String.

Output: CorrespondenceList.

The Correspondence Detector component compares input utterance with the SignaturePart of all FeatureCouplet(s) (and possibly MetaCouplet(s)) stored in the FeatureCouplet (and MetaCouplet) database(s). Their address(es), along with their LocalCorrespondence and GlobalCorrespondence ratios, are output to the CorrespondenceList.

6.4.1.2 The FuzzyCorrespondence detector as implemented in FuzzBabe:

Input: any String.

Output: FuzzyCorrespondenceList

The FuzzyCorrespondence Detector component compares input utterance with the SignaturePart of all FeatureCouplet(s) (and possibly MetaCouplet(s)) stored in the FeatureCouplet (and MetaCouplet) database(s). Their address(es), along with their LocalFuzzyCorrespondence and GlobalFuzzyCorrespondence ratios, are output to the FuzzyCorrespondenceList.

6.4.1.3 The FuzzyInference component.

Input: the CorrespondenceList or the FuzzyCorrespondenceList

Output: the InferredList.

The FuzzyInference component performs max-product evaluation on local and global correspondence metrics for each Feature contained in the CorrespondenceList or the FuzzyCorrespondenceList and outputs the address(es) of InferredCouplets along with their Correspondence or FuzzyCorrespondence ratios to the InferredList.

6.4.1.4 The Defuzzification component.

Input: the InferredList

Output: the PrimaryCrispList.

The Defuzzification component takes the InferredList containing the address of each Feature along with their Correspondence or FuzzyCorrespondence ratios, and performs max-membership principle evaluation on data to produce PrimaryActivatedCouplets. The component outputs the address(es) of PrimaryActivatedCouplets along with their Correspondence or FuzzyCorrespondence ratios to the PrimaryCrispList.

6.4.1.5 Process flow diagram (Pattern Comparator as implemented in CorrBabe).

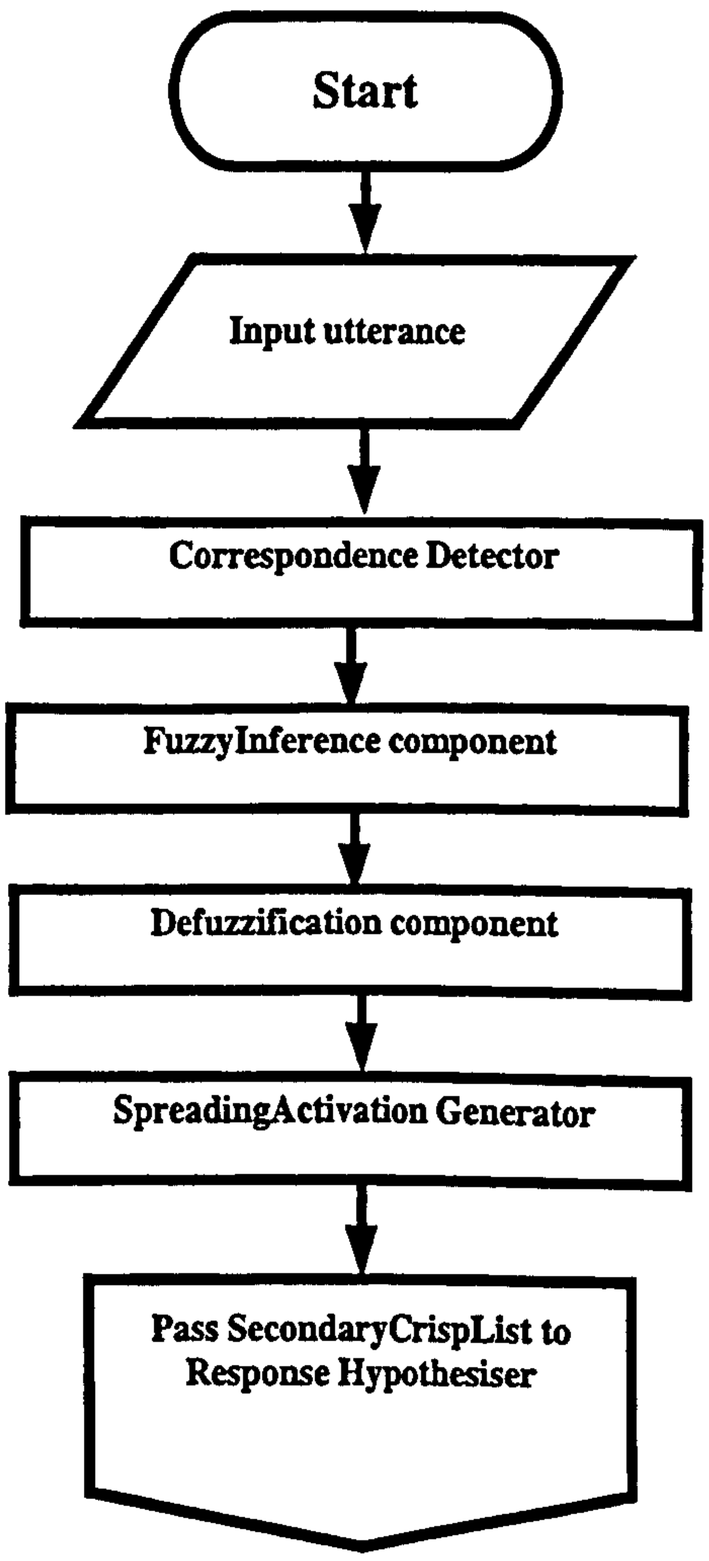


Figure 11: Process flow diagram of Pattern Comparator module as implemented in CorrBabe.

6.4.1.6 Process flow diagram. (Pattern Comparator as implemented in FuzzBabe)

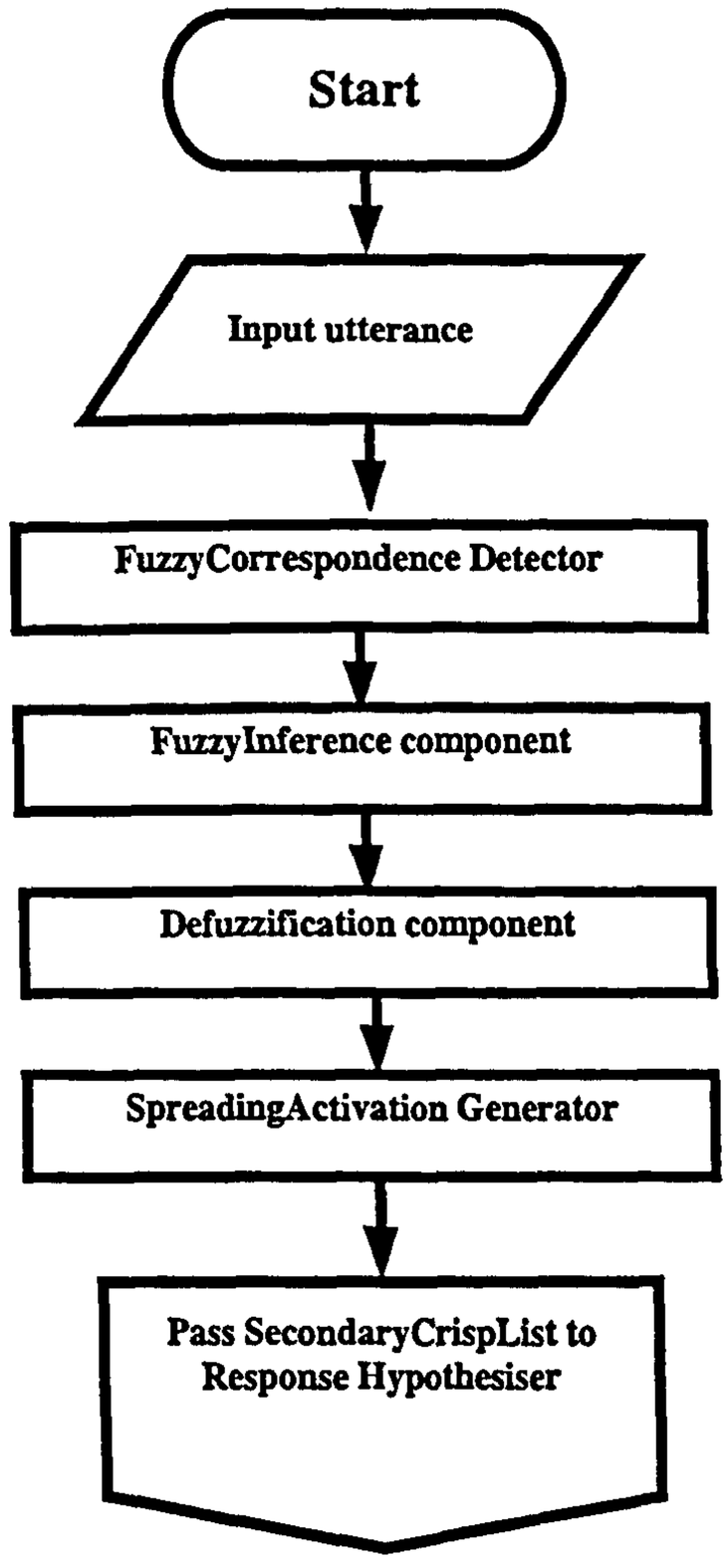


Figure 12: Process flow diagram of Pattern Comparator module as implemented in FuzzBabe.

6.5 Systems Data flow diagram.

There follows two diagrams that shows the data flow within the CorrBabe and FuzzBabe systems. The iconic convention employed is the same as that used for NonfuzzBabe:

Figure 13: CorrBabe Data Flow Diagram

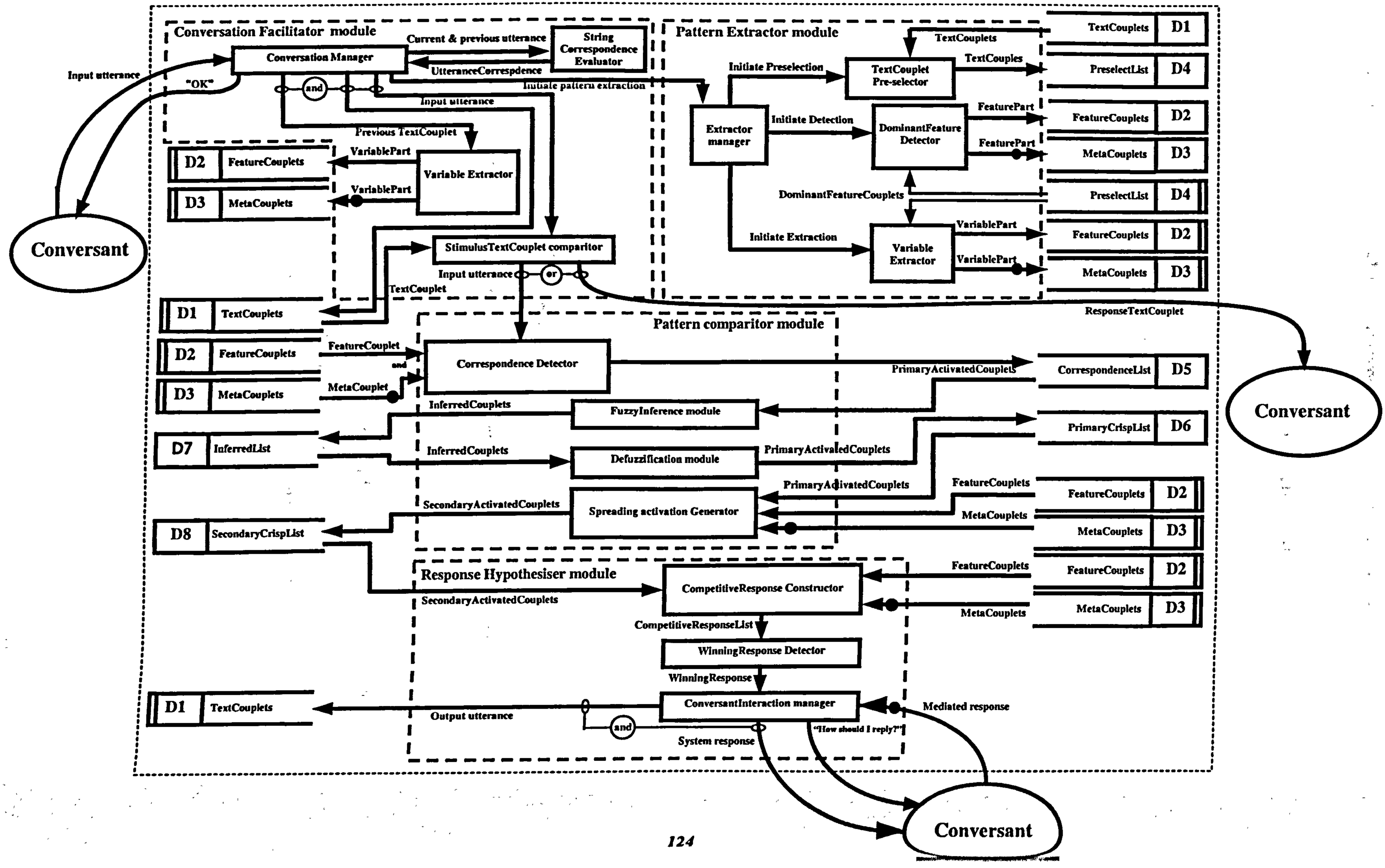
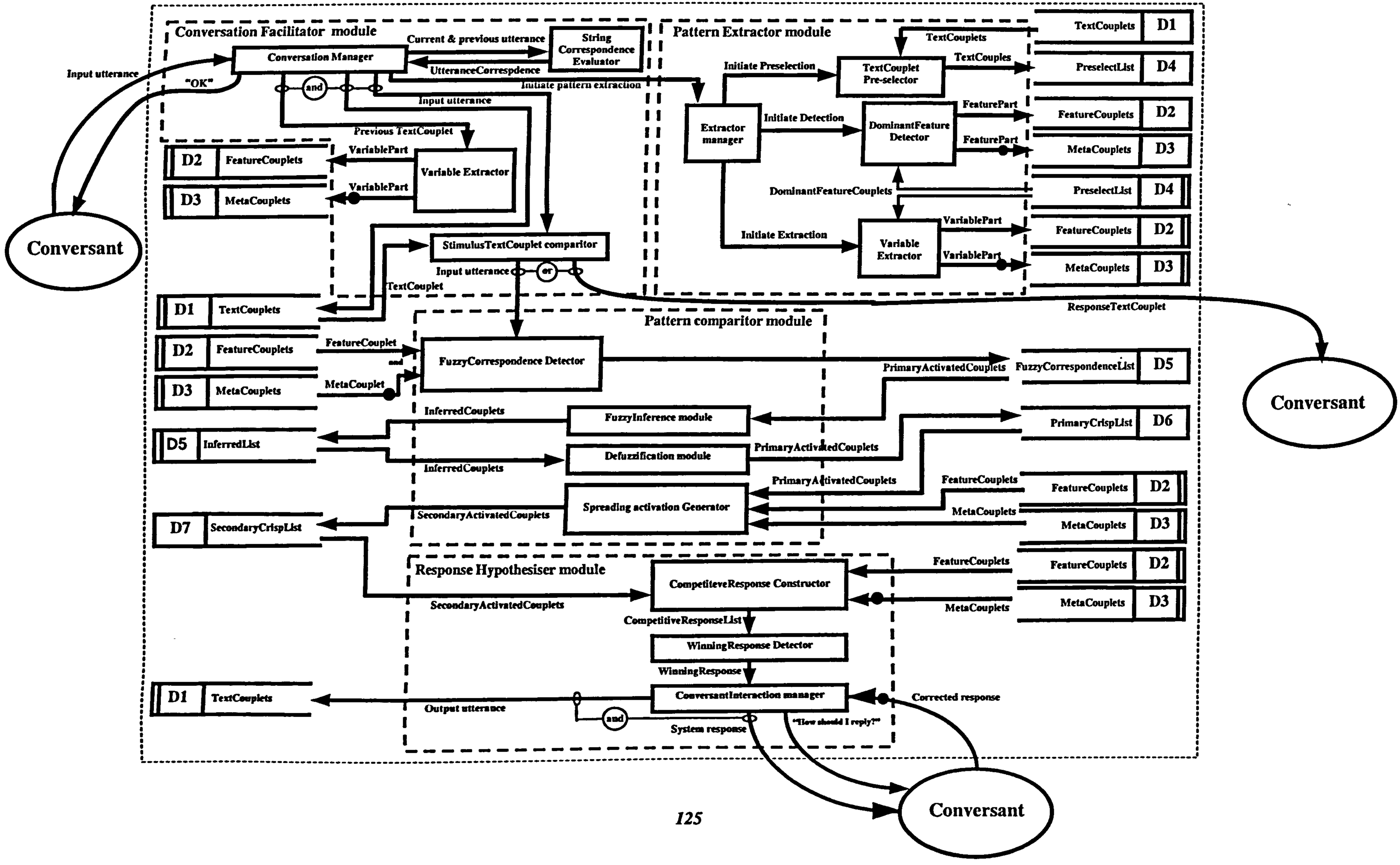


Figure 14: FuzzBabe Data Flow Diagram



6.6 Data dictionaries for CorrBabe and FuzzBabe.

6.6.1 Data dictionary for CorrBabe.

No.	DFD Name (entity name)	Name in VB	Data Type	Variable Type	Data description
D1	TextCouplets	lstdb(0)	String	List	Input utterances that either: a) have not been input to the Pattern Extraction process; b) share insufficient commonality to form FeatureCouplets.
D2	FeatureCouplets	lstdb(1)	String	List	Output of the Pattern Extraction process, input with TextCouplets.
D3	MetaCouplets	lstdb(2) - lstdb(7)	String	List	Output of the Pattern Extraction process, input with FeatureCouplets or MetaCouplets
D4	Pre-selectList	Buffer 1	String	List	DominantFeatureCouplets
D5	CorrespondenceList	List1(0) – List1(3)	Single	List Array	List1(0) - Base location of SecondaryActivatedCouplets. List1(1) - List location of SecondaryActivatedCouplets. List1(2) – LocalCorrespondence ratio List1(3) – GlobalCorrespondence ratio
D6	PrimaryCrispList	List7(0) – List7(2)	Single	List Array	List7(0) – Base location of SecondaryActivatedCouplets. List7(1) – List location of SecondaryActivatedCouplets. List7(2) – Fuzzy dot product evaluation
D7	InferredList	List8(0) – List8(2)	Single	List Array	List8(0) – Base location of SecondaryActivatedCouplets. List8(1) – List location of SecondaryActivatedCouplets. List8(2) – Max-product evaluation
D8	SecondaryCrispList	List12(0) – List12(2)	Single	List Array	List12(0) – Base location of SecondaryActivatedCouplets. List12(1) – List location of SecondaryActivatedCouplets. List12(2) – max-membership principle evaluation

Table 12: Data dictionary for CorrBabe.

6.6.2 Data dictionary for FuzzBabe.

No.	DFD Name (entity name)	Name in VB	Data Type	Variable Type	Data description
D1	TextCouplets	lstdb(0)	String	List	Input utterances that either: a) have not been input to the Pattern Extraction process; b) share insufficient commonality to form FeatureCouplets.
D2	FeatureCouplets	lstdb(1)	String	List	Output of the Pattern Extraction process, input with TextCouplets.
D3	MetaCouplets	lstdb(2) - lstdb(7)	String	List	Output of the Pattern Extraction process, input with FeatureCouplets or MetaCouplets
D4	Pre-selectList	Buffer 1	String	List	DominantFeatureCouplets
D5	FuzzyCorrespondenceList	List1(0) – List1(3)	Single	List Array	List1(0) - Base location of SecondaryActivatedCouplets. List1(1) - List location of SecondaryActivatedCouplets. List1(2) – LocalCorrespondence ratio List1(3) – GlobalCorrespondence ratio
D6	PrimaryCrispList	List7(0) – List7(2)	Single	List Array	List7(0) – Base location of SecondaryActivatedCouplets. List7(1) – List location of SecondaryActivatedCouplets. List7(2) – Fuzzy dot product evaluation
D7	InferredList	List8(0) – List8(2)	Single	List Array	List8(0) – Base location of SecondaryActivatedCouplets. List8(1) – List location of SecondaryActivatedCouplets. List8(2) – Max-product evaluation
D8	SecondaryCrispList	List12(0) – List12(2)	Single	List Array	List12(0) – Base location of SecondaryActivatedCouplets. List12(1) – List location of SecondaryActivatedCouplets. List12(2) – max-membership principle evaluation

Table 13: Data dictionary for FuzzBabe.

PAGE NUMBERING AS ORIGINAL

6.7 Summary.

In this chapter, the CorrBabe and FuzzBabe systems have been described in some detail. Firstly, the rationale that underpins implementations is reviewed, then new terms are defined. This was followed by a short exposé concerning fuzzy processing in general. An account was then given concerning the way in which fuzzy processing is applied in Baby models. An overview of systems behaviours was then given, followed by a separate specification for each of the modules that differed from the NonfuzzBabe implementation. Finally data flow diagrams and data dictionaries were presented for each system. For program specific levels of description, appendix A contains the appropriate code.

7.0 Language performance of NonfuzzBabe, CorrBabe and FuzzBabe.

This chapter shows results that were obtained from some experiments conducted upon NonfuzzBabe, CorrBabe and FuzzBabe in order to provide data to test the second hypothesis stated in chapter *3.0 Research strategy.*, i.e. that:

> Baby models, implemented with fuzzy classification techniques, will produce superior simple language processing performance compared with Baby models implemented using equivalence classification techniques, as judged by a human user of the language employed.

Set out below is an overview of the experimental work that has been carried out to test this hypothesis:

1. Both CorrBabe and FuzzBabe were applied to the same tests that were applied to NonfuzzBabe as detailed in chapter *5.0 Results of NonfuzzBabe's language performance.*

2. It will be seen that encouraging results were obtained with these tests and it was therefore decided to explore the comparative performance of NonfuzzBabe, CorrBabe and FuzzBabe further. One criticism of the tests applied to NonfuzzBabe, is that because both training and test sets are small, they may not provide a representative sample of each syntactic transformation tested. It was therefore decided to conduct an in-depth evaluation of one of the simple syntactic transformations, that of past tense formation.

In addition, two further issues will be addressed:

3. Other researchers have investigated machine acquisition of past tense formation. The results from five such experiments will be compared with the Baby model.

4. During this investigation, many interesting Baby behaviours were observed and one has been chosen to include in this chapter, i.e. the ability of the Baby model to process textual representations of language containing no white-space.

Results for each experiment are given, however, design and set-up are described only for experiments associated with 2 above. Description and analysis of the results appear later in section *7.5 Discussion and Analysis.* The chapter concludes with a summary of issues covered.

7.1 Results of comparative tests with NonfuzzBabe.

This section shows results obtained by applying CorrBabe and FuzzBabe to the tests applied to NonfuzzBabe, as described in chapter *5.0 Results of NonfuzzBabe's language performance.* The reader is referred to that chapter for details of experimental design and set-up. Table 14 and Table 15 show the results obtained with CorrBabe and FuzzBabe, and Table 16 shows comparative data between all three implementations.

7.1.1 CorrBabe results.

Syntactic transformation test	Main comments	Accuracy
Subject verb agreement	Both transformation rules learnt. Exception learnt.	100%
Past tense formation	Three of three transformation rules learnt. Exception learnt.	100%
Plural formation	Two of two transformation rules learnt. Exception not learnt.	80%
Bi-lingual transformation	Two of two transformation rules learnt. Exception not learnt.	80%
Passive transformation	Two of two transformation rules learnt. Exception not learnt.	86%
Sensitivity	Six of six transformation rules learnt. No exception tested.	100%
Selectivity	Four of four transformation rules learnt. No exception tested.	100%

Table 14: Accuracy performance of CorrBabe when applied to simple syntactic transformation tests.

7.1.2 FuzzBabe results.

Syntactic transformation test	Main comments	Accuracy
Subject verb agreement	Both transformation rules learnt. Exception learnt.	100%
Past tense formation	Three of three transformation rules learnt. Exception learnt.	100%
Plural formation	Two of two transformation rules learnt. Exception not learnt.	80%
Bi-lingual transformation	Two of two transformation rules learnt. Exception not learnt.	80%
Passive transformation	Two of two transformation rules learnt. Exception not learnt.	86%
Sensitivity	Six of six transformation rules learnt. No exception tested.	100%
Selectivity	Four of four transformation rules learnt. No exception tested.	100%

Table 15: Accuracy performance of FuzzBabe when applied to simple syntactic transformation tests.

7.1.3 Accuracy comparison between NonfuzzBabe, CorrBabe and FuzzBabe.

Syntactic transformation test	NonfuzzBabe	CorrBabe	FuzzBabe
Subject verb agreement	60%	100%	100%
Past tense formation	43%	100%	100%
Plural formation	80%	80%	80%
Bi-lingual transformation	0%	80%	80%
Passive transformation	86%	86%	86%
Sensitivity	83%	100%	100%
Selectivity	100%	100%	100%

Table 16: Accuracy performance of NonfuzzBabe, CorrBabe and FuzzBabe when applied to simple syntactic transformation tests.

7.2 Results of comparative tests using larger training and test corpora.

This section describes experimental design and set-up, then shows results obtained by applying NonfuzzBabe, CorrBabe and FuzzBabe to larger, randomly selected training and test sets of the simple syntactic transformation of past tense formation.

7.2.1 Experimental design.

The experiment was designed to test NonfuzzBabe, CorrBabe and FuzzBabe in three aspects of past tense learning:

- training accuracy;
- seen verb prediction accuracy;
- unseen verb prediction accuracy.

The experiment was designed to gather data that would indicate two aspects of Baby models. Firstly, an absolute indication of training and prediction accuracy and secondly, an indication of how prediction accuracy varies with different sized training corpora. This latter data is useful because language acquisition models are often tested in this way and consequently some comparative data with other systems will emerge (see section *7.3 Results of comparative tests with other research models.*). The verb corpus used in this set of experiments was compiled by Prof. Brian MacWhinney and is used (and sections reprinted here) with his permission (MacWhinney, 1999). A copy of the MacWhinney corpus is shown in appendix B. The corpus was chosen, primarily because it was employed by one of the systems with which Baby models are compared, though other issues such as language and corpus size also played a part in the choice. Amongst other information, the corpus

contains about 1200 verbs with their regular past tense formation. Both training and test corpora are generated from this verb corpus and consequently the maximum size for the training corpus was set at 1000 verbs thus leaving 200 for test corpus purposes. To test how accuracy varied with different sized training corpora, 10 other sized training corpora where chosen, 500, 300, 200, 100, 50, 30, 20, 10, 5 and 2. Test corpora used for training accuracy tests was the training corpus itself, and the size of test corpora used for prediction accuracy tests was 30 in all cases.

Most previous research into machine learning of past tense formation has been conducted using only the verb and its past tense formation in both training and test sets. It was decided to adopt this practice here. The purpose of this research, however, is to investigate the processing of character patterns occurring in textual representations of language. In addition to verb-only tests, it was therefore decided to use training and test corpora containing interactional discourse representations of past tense formation, (see page 149 for further discussion on this issue). The conversational format used was:

> **What is the past tense of** ***[base verb]*****?**
>
> **The past tense of** ***[base verb]*** **is** ***[past tense of base verb]*****.**

The experiment comprises six parts, each part conducted with each of the different sized training corpora as follows:

1) training accuracy (sentential structure);
2) training accuracy (verb-only);
3) prediction accuracy with seen verbs (sentential structure);
4) prediction accuracy with seen verbs (verb-only);
5) prediction accuracy with unseen verbs (sentential structure);
6) prediction accuracy with unseen verbs (verb-only).

To reduce bias and thus increase reliability, each set of these six tests was run on three different occasions, each time using a different randomisation of the MacWhinney corpus.

The total experiment entailed the application of each of NonfuzzBabe, CorrBabe and FuzzBabe to the six parts shown above, each part conducted with different sized corpora as detailed below. Each of these sets of test was conducted on 3 separate occasions, each time using a different randomly selected corpus. This comprised a total of 522 tests. In all, Baby models were tested with some 40,000 utterances.

7.2.2 Experimental set up.

The MacWhinney corpus contains 2163 base verbs, each annotated with various tenses. 1394 of all base verbs are annotated with their past tense, and 1258 of these verbs form a regular past tense and 136 form an irregular past tense. In order to efficiently extract and randomise information from the corpus, a special purpose *corpus extractor tool* was designed, implemented and tested. The corpus extractor was implemented in Visual Basic 5. Details of the implementation of this software are not given in this thesis.

Firstly, all base verbs that form regular past tenses were extracted along with their past tense into a list. Members of this list were then chosen randomly without replacement and stored in a RandomList. The RandomList was then used to produce four corpora, two verb-only corpora and two sentential structure corpora as follows:

1. the first 1000 base verbs each followed by its past tense;
2. the first 1000 base verbs with each of their past tense, embedded into two sentential structures;
3. the 1001-1030 base verbs each followed by its past tense;
4. the 1001-1030 base verbs with each of their past tense, embedded into two sentential structures.

Corpora 1 and 2 form the training sets used respectively in the verb-only and sentential structure tests. Smaller training sets were derived from corpora 1 and 2 by extracting the $1^{st.}$ to $n^{th.}$ members, e.g. 200 training corpora were derived from the first 200 members of corpora 1 and 2 etc.

Corpora 3 and 4 were used respectively to test verb-only and sentential structure conditions when testing unseen verbs. Seen verb test corpora were taken from the first 30 verbs of the 1000 corpus for both sentential and verb-only tests.

To reduce bias and thus increase reliability, each of the above tests was run on three different occasions. On each occasion, corpora 1 - 4 were compiled from a different randomisation of the regular verb list derived from the Whitney corpus.

7.2.3 Summary of results obtained.

There follows a set of tables that summarise the performance of NonfuzzBabe, CorrBabe and FuzzBabe in the six test conditions specified in sub-section *7.2.1 Experimental design.* The tables show the percentage accuracy achieved by each implementation for each sized training corpus for each of the six tests. The summary is followed by a graphical representation of the data in each test condition.

7.2.3.1 Training accuracy % (sentential).

Training corpus / Process method	1000	500	300	200	100	50	30	20	10	5	2
NonfuzzBabe	55	54	57	58	55	55	58	65	50	46	66
CorrBabe	69	68	71	70	73	70	64	76	60	60	66
FuzzBabe	70	70	72	71	74	71	65	76	63	60	66

Table 17: Training accuracy % (sentential embedding)

7.2.3.2 Training accuracy % (verb-only).

Training corpus / Process method	1000	500	300	200	100	50	30	20	10	5	2
NonfuzzBabe	74	65	58	59	60	54	35	56	36	26	33
CorrBabe	71	66	64	64	59	49	37	51	36	26	33
FuzzBabe	74	68	65	64	58	50	35	50	36	26	33

Table 18: Training accuracy % (verb-only)

7.2.3.3 Prediction accuracy % with seen verbs (sentential).

Training corpus / Process method	1000	500	300	200	100	50	30	20	10	5	2
NonfuzzBabe	50	51	53	53	52	51	58				
CorrBabe	74	66	73	66	66	67	64				
FuzzBabe	77	67	74	67	67	67	65				

Table 19: Prediction accuracy % with seen verbs (sentential embedding).

7.2.3.4 Prediction accuracy % with seen verbs (verb-only).

Training corpus / Process method	1000	500	300	200	100	50	30	20	10	5	2
NonfuzzBabe	67	57	56	61	57	51	35				
CorrBabe	73	64	63	65	52	46	37				
FuzzBabe	74	62	63	65	54	47	34				

Table 20: Prediction accuracy % with seen verbs (verb-only).

7.2.3.5 Prediction accuracy % with unseen verbs (sentential).

Training corpus / Process method	1000	500	300	200	100	50	30	20	10	5	2
NonfuzzBabe	57	55	56	54	56	58	54	57	58	43	45
CorrBabe	56	51	52	52	52	60	66	75	71	56	47
FuzzBabe	64	60	55	56	53	64	72	78	73	56	47

Table 21: Prediction accuracy % with unseen verbs (sentential embedding).

7.2.3.6 Prediction accuracy % with unseen verbs (verb-only).

Training corpus / Process method	1000	500	300	200	100	50	30	20	10	5	2
NonfuzzBabe	58	58	38	52	35	25	14	26	8	1	2
CorrBabe	38	42	30	34	21	17	13	22	8	1	6
FuzzBabe	45	42	31	38	25	20	13	23	8	1	6

Table 22: Predication accuracy % with unseen verbs (verb-only).

There follows a graphical representation of the data shown in the above tables.

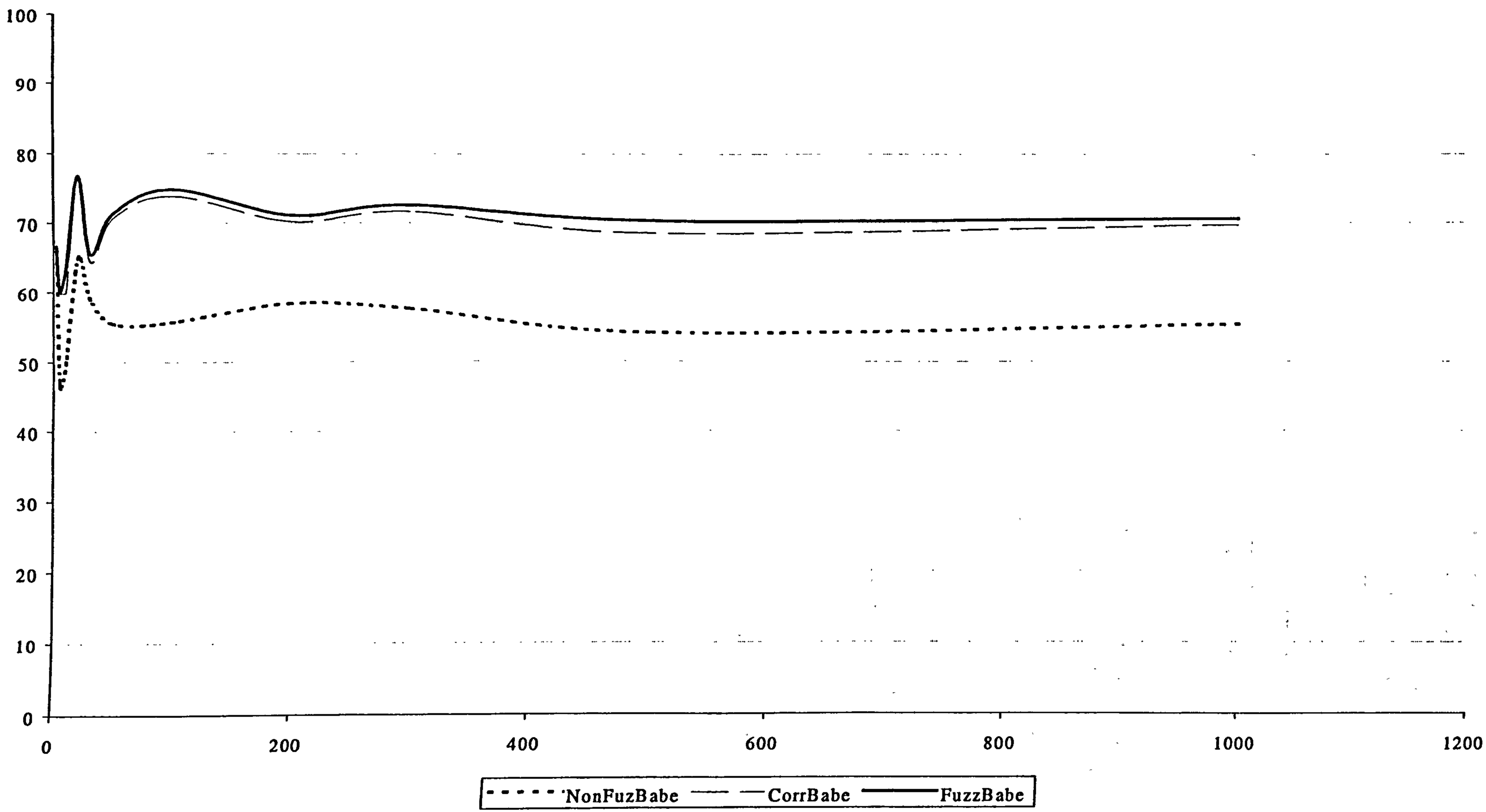


Figure 15: *Training accuracy % vs. corpus size (sentential embedding).*

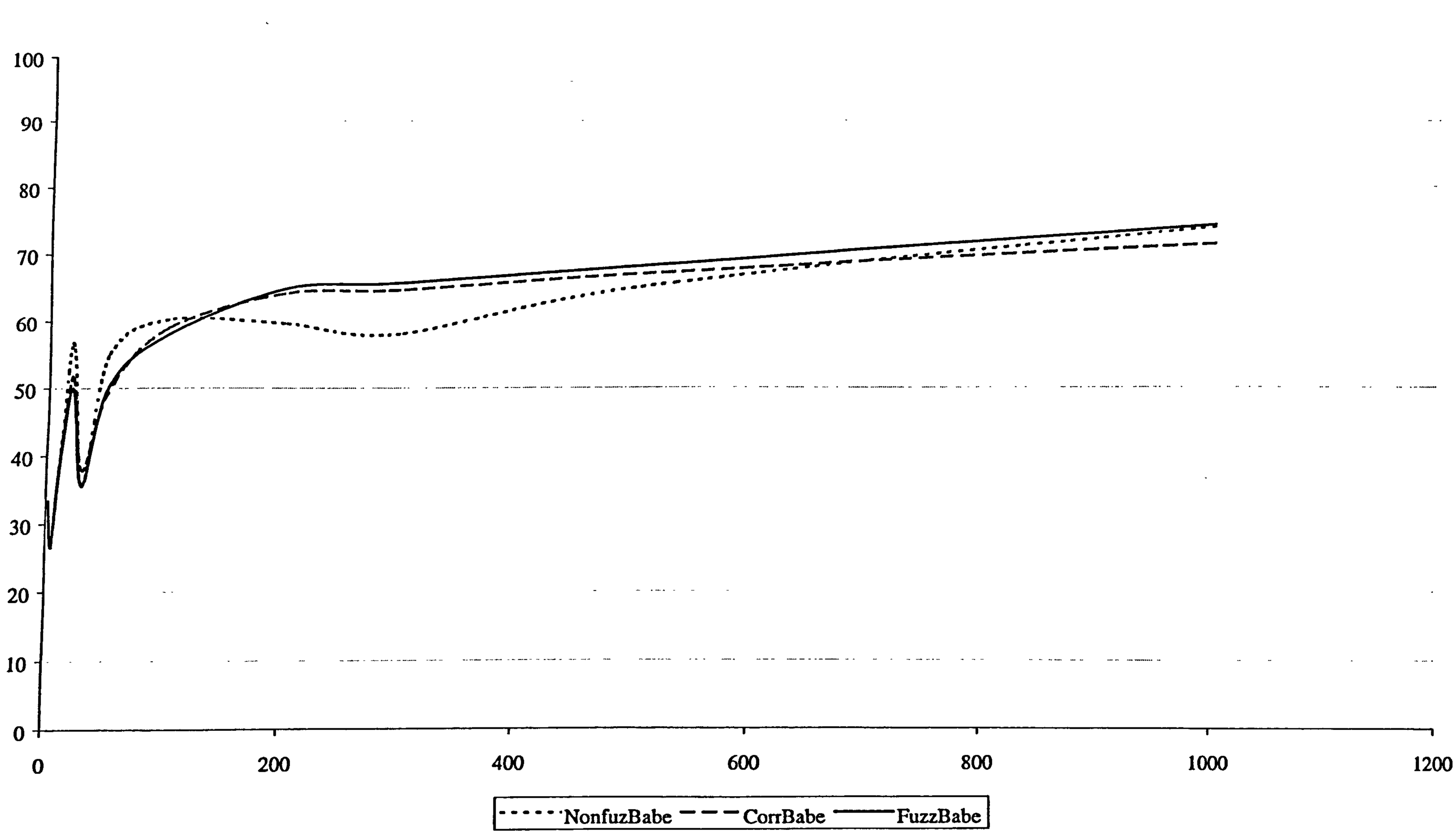


Figure 16: Training accuracy % vs. corpus size (verb only).

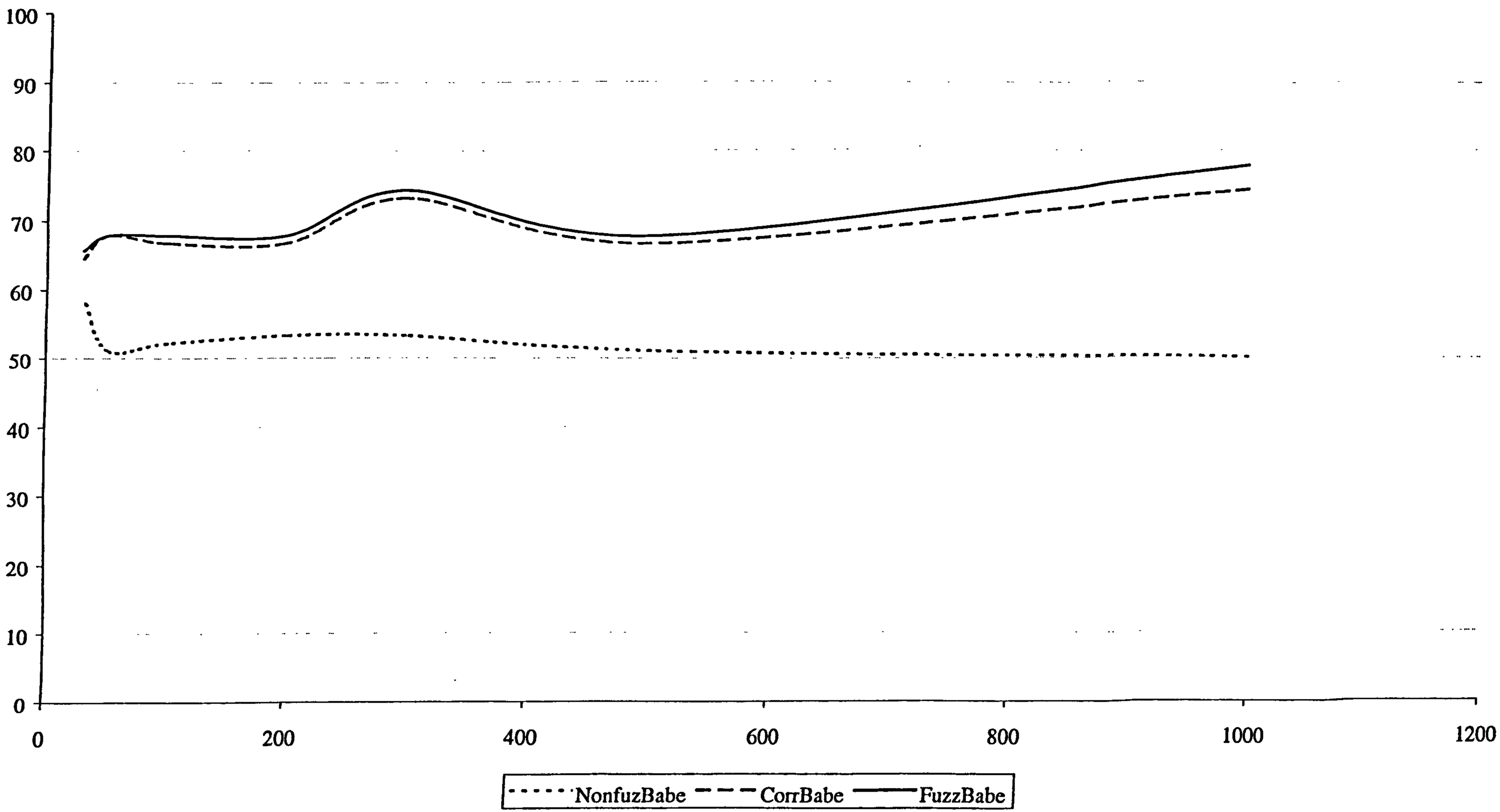


Figure 17: Prediction accuracy % with seen verb vs. training corpus size (sentential embedding).

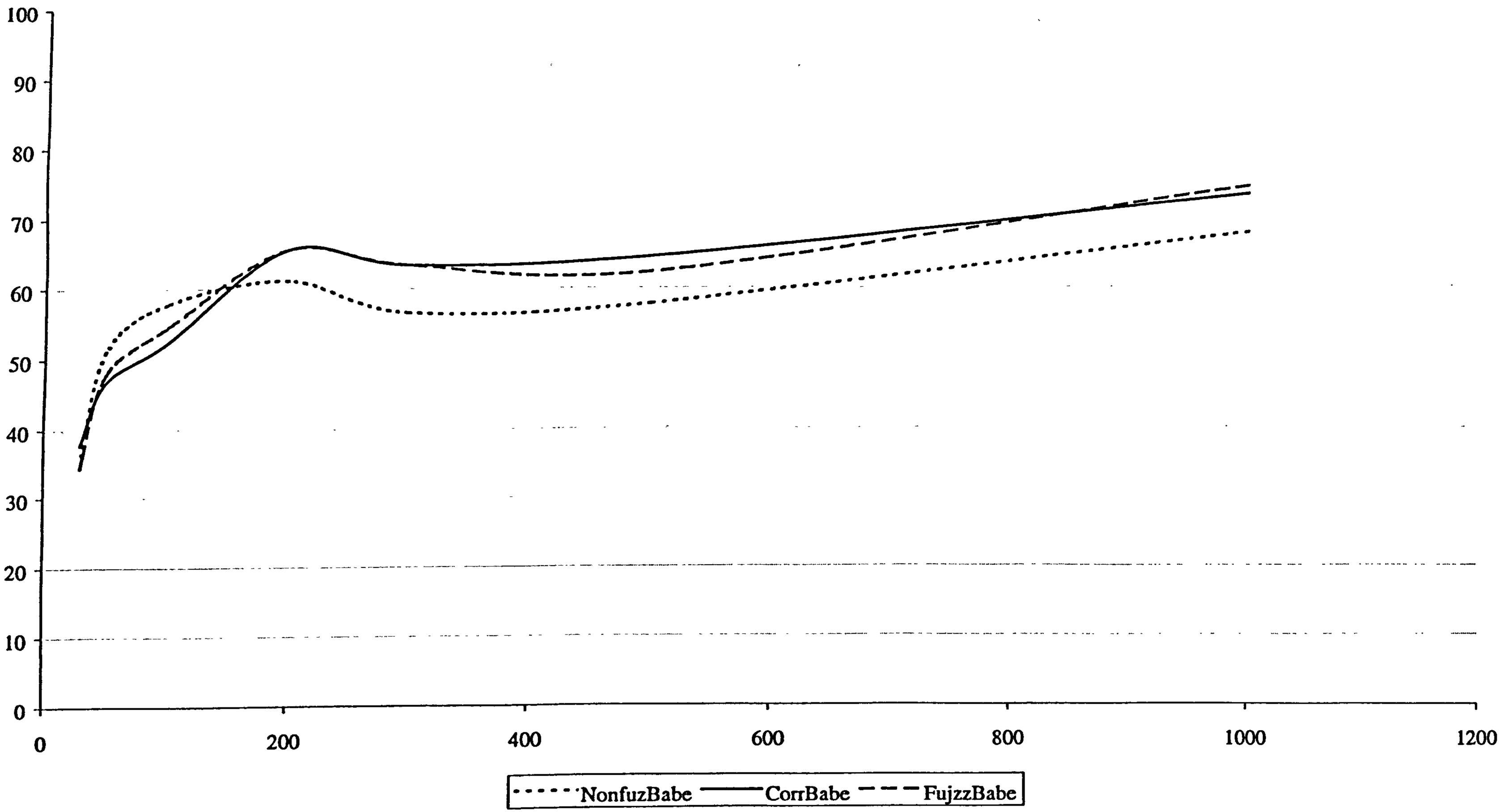


Figure 18: Prediction accuracy % with seen verb vs. training corpus size (verb only).

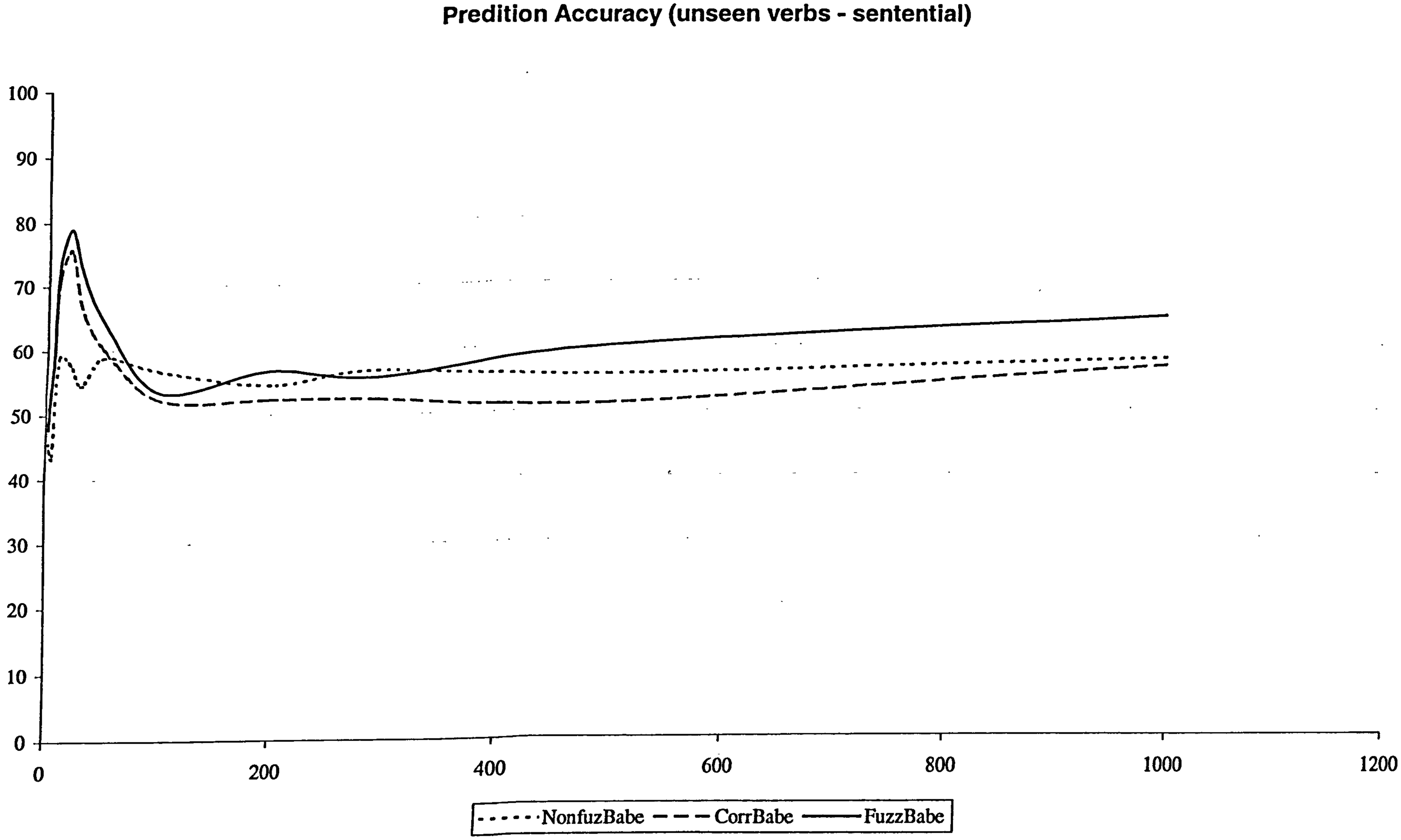


Figure 19: *Prediction accuracy % with unseen verbs vs. training corpus size (sentential embedding).*

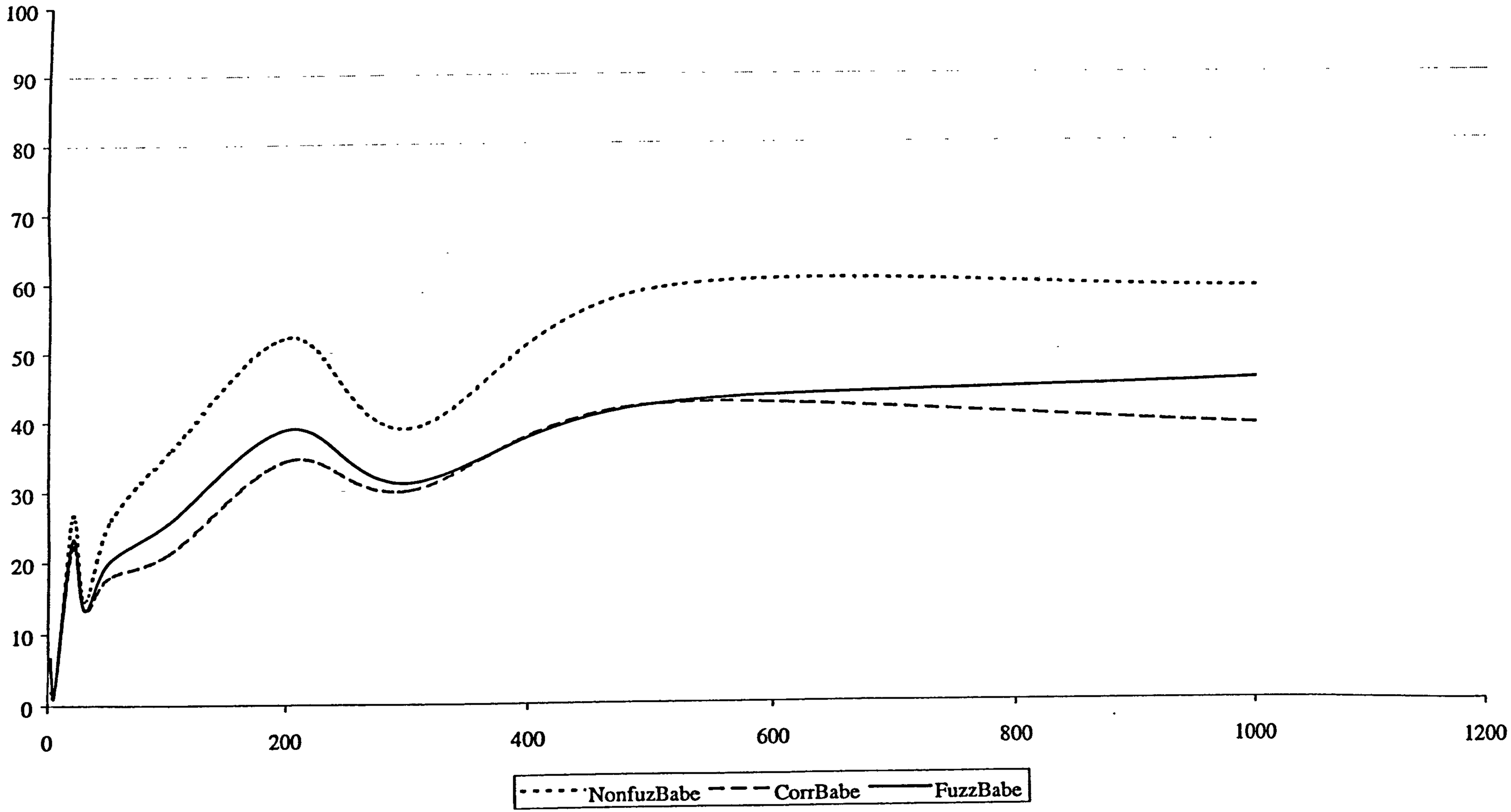


Figure 20: Prediction accuracy % with unseen verbs vs. training corpus size (verb only).

PAGE NUMBERING AS ORIGINAL

7.3 Results of comparative tests with other research models.

In this section, results are shown that compare the performance between five past tense acquisition models produced by other researchers and the Baby model. Each of these other research models acquire past tense formation using verb-only training and test corpora. For each Baby model, comparative data is presented showing percentage accuracy when applied to verb-only and also sentential training and test corpora.

7.3.1 The Rumelhart and McClelland model.

Table 23 compares NonfuzzBabe, CorrBabe and FuzzBabe to the performance of a model presented in Rumelhart and McClelland (1986). They describe an implementation that is exposed to verb-only training and test corpora. Baby data is extracted from data contained in *7.2 Results of comparative tests using larger training and test corpora.*

	McRum	NonfuzzBabe	CorrBabe	FuzzBabe
Verb-only	66	58	42	42
Sentential structure	N/A	55	51	60

Table 23: Comparison between Rumelhart & McClelland model and NonfuzzBabe, CorrBabe and FuzzBabe.

7.3.2 The Ling models.

Table 24 shows data comparing four models presented in Ling (1994), with NonfuzzBabe, CorrBabe and FuzzBabe. Each of Ling's tests use training and test corpora containing verb-only present and past tense constructs. Baby data is extracted from data contained in *7.2 Results of comparative tests using larger training and test corpora.*

Training size	SPA (adaptive)	SPA (majority)	ANN (copy con.)	ANN (normal)	NonfuzzBabe	CorrBabe	FuzzBabe
50	55	30	14	7	25	17	20
100	72	58	34	24	35	21	25
300	87	83	59	58	38	30	31
500	92	89	82	67	58	42	42
1000	93	92	92	87	58	38	45

Table 24: Comparison between Ling models and NonfuzBabe, CorrBabe and FuzzBabe using verb-only.

Table 25 shows data comparing the four Ling models with NonfuzBabe, CorrBabe and FuzzBabe. Test conditions for the Ling models remain identical to those shown in Table 24, whereas Baby implementations are exposed to present and past tense embedded in sentential structure, constructs. Baby data is extracted from data contained i*n 7.2 Results of comparative tests using larger training and test corpora.*

Training size	SPA (adaptive)	SPA (majority)	ANN (copy con.)	ANN (normal)	NonfuzzBabe	CorrBabe	FuzzBabe
50	55	30	14	7	58	60	64
100	72	58	34	24	56	52	53
300	87	83	59	58	56	52	55
500	92	89	82	67	55	51	60
1000	93	92	92	87	57	56	64

Table 25: Comparison between Ling models and NonfuzzBabe, CorrBabe and FuzzBabe using sentential embedding.

7.4 Results of processing text with no white-space.

Table 26 shows results that were obtained from FuzzBabe when applied to the simple syntactic transformation of pluralisation. The training and test corpora are the same as those used in the pluralisation test in *7.1 Results of comparative tests with NonfuzzBabe.* i.e. using the general format:

Whatisthepluralof*[base verb]***?**

Thepluralof *[base verb]***is***[plural tense of base verb]***.**

No.	Human utterance	FuzzBabe's reply
1a	Whatisthepluralofsnake?	
1b	Thepluralofsnakeissnakes.	
2a	Whatisthepluralofcow?	
2b	Thepluralofcowiscows.	
3a	Whatisthepluralofdatum?	
3b	Thepluralofdatumisdata.	
4a	Whatisthepluralofagendum?	
4b	Thepluralofagendumisagenda.	
5a	Whatisthepluralofdeer?	
5b	Thepluralofdeerisdeer.	
	Pattern Extraction Initiated	
6	Whatisthepluralofsnake?	Thepluralofsnakeissnakes.
7	Whatisthepluralofnet?	Thepluralofnetisnets.
8	Whatisthepluralofagendum?	Thepluralofagendumisagenda.
9	Whatisthepluralofseptum?	Thepluralofseptumissepta.
10	Whatisthepluralofdeer?	Thepluralofdeerisdeers.

Table 26: Results of test that illustrates the processing of textual representations of language from which white-space has been removed.

7.5 Discussion and Analysis.

This section contains a discussion and analysis of results presented. The objective was to test three aspects of the performance of NonfuzzBabe, CorrBabe and FuzzBabe:

- the performance of CorrBabe and FuzzBabe when using the simple syntactic transformation tests applied to NonfuzzBabe detailed in chapter *5.0 Results of NonfuzzBabe's language performance.*;
- their performance when applied to representative training and test corpora;
- their performance compared with other researcher's models.

In addition to these tests, opportunity is taken in this chapter to illustrate an interesting psychologically plausible Baby behaviour. The experiments presented in sections 10.1, 10.2, 10.3 and 10.4 respectively address each of these issues.

7.5.1 Comparative test with NonfuzzBabe.

In section *2.10.7 Analysis of formalisms capable of processing nonmonotonic problem spaces.*, it was argued that a fuzzy formalism was best suited to model the concept represented by "most closely correspond" in the theory underpinning the Baby concept. Fuzzy mechanisms, however, can exhibit inaccuracy and/or overproductivity and these negative attributes may degrade, language performance compared with NonfuzzBabe. The purpose of this test is to gain evidence for or against. Table 16 requires little analysis as the outcome is evident. Of the seven simple syntactic transformation tests applied to NonfuzzBabe, both CorrBabe and FuzzBabe equalled the performance of NonfuzzBabe on three tests, and exceeded NonfuzzBabe performance on the other four tests. Indeed in the case of the bi-lingual transformation NonfuzzBabe was unable to process any correct responses, whereas CorrBabe and FuzzBabe both achieved 80% accuracy. Compared with NonfuzzBabe, the average improvement in performance across all seven tests with CorrBabe and FuzzBabe was 27%. It was noted that the performance of CorrBabe and FuzzBabe was identical for all seven tests. This may indicate that no practical difference exists between the correspondence and FuzzyCorrespondence classification methods. It was considered necessary to investigate this phenomenon further, however. It was suspected that it may be due to small number of patterns extracted from the small training corpora used. Within the domain of the test requirements, no criticism could be found for the tests. In view of the above analysis it was decided to continue with further experimentation.

7.5.2 Comparative tests using larger training and test corpora.

From a wider perspective, a criticism of the tests applied to NonfuzzBabe is that because both training and test sets are relatively small and contrived, they probably do not provide a representative sample of each syntactic transformation tested. These tests cannot, therefore, be considered a serious indicator of the Baby's natural language performance, rather they indicate Baby functionality. The simple tests also showed no difference in performance between CorrBabe and FuzzBabe performance. The tests analysed in this section address these issues. To conduct experiments containing representative training and test sets on all of the simple syntactic transformation types applied in the basic tests, was considered to be too large an exercise for this research project. It was therefore decided to conduct an in-depth evaluation of one of the simple syntactic transformations, that of past tense formation. The method chosen to overcome the limited exposure to this behaviour in the NonfuzzBabe tests, was to significantly increase the size of both training and test sets, and to conduct multi-runs and average results.

Forming the past tense of verbs appears a minor aspect of natural language. Configuring machines that acquire such language behaviour, however, has generated interest, comment and research since early connectionist implementations. "*.....learning the past tense* [of English verbs] *has become a landmark task for testing the adequacy of cognitive modelling.*", (Ling 1994). Models have been developed within both connectionist and symbolic paradigms. Landmark connectionist models are Rumelhart and McClelland (1986) and MacWhinney and Leinbach (1991). Symbolic implementations often use a Symbolic Pattern Associator, based upon Quinlan, (1993). Such implementations are exemplified in Ling (1994)

The graphical representation of NonfuzzBabe, CorrBabe and FuzzBabe performance (Figure 15, Figure 16, Figure 17, Figure 18, Figure 19 and Figure 20) shows some interesting behaviour. With the exception of training accuracy - sentential (Figure 15), that possesses a relatively flat response, all other graphs show a positive correlation between training corpus size and accuracy. In addition, with the exception of prediction accuracy using unseen verbs - verb-only (Figure 20), a positive correlation is observable between accuracy and increasing degrees of fuzzy classification.

All graphs illustrate that the function relating accuracy and test corpus size is more complex than is usual for functions that represent the same behaviour in more traditional approaches. These discontinuities manifest in two distinct ways. Firstly, growth rate using larger test and/or training sets is not curvi-linear and secondly chaotic behaviour is observable with small test and/or training sets. These effects are more noticeable with the prediction of unseen verb tests. Graphical representations showing the behaviour of individual runs within the averaged three run data set show similar small test and/or training set behaviour, but significantly increased non curvi-linear

relationships with larger test and/or training. It is hypothesised that the chaotic performance may be a real system behaviour whilst fluctuations with larger test and/or training sets will eventually average out. This hypothesis has not been fully investigated; however averaged results from many different runs, all using small test and/or training sets, have shown similar chaotic behaviour. The consequences of this hypothesis being true are discussed in the conclusion of this thesis. The phenomenon was noticed early in research and is the reason that training and test set sizes include sets that are much smaller than those normally employed to test more traditional approaches.

Table 27 shows the average performance of NonfuzzBabe, CorrBabe and FuzzBabe for three experimental runs over all sized training sets. The data is presented for each experimental condition.

Test number	**1**	**2**	**3**	**4**	**5**	**6**
Test / Implementation	**Training** (sentential)	**Training** (verb-only)	**Prediction** seen verbs (sentential)	**Prediction** seen verbs (verb-only)	**Prediction** unseen verbs (sentential)	**Prediction** unseen verbs (verb-only)
NonfuzzBabe	56.68	50.96	52.86	55.40	54.54	29.39
CorrBabe	68.37	51.06	68.57	57.62	58.38	21.51
FuzzBabe	69.32	51.19	69.84	57.46	62.16	23.33

Table 27: NonfuzzBabe, CorrBabe and FuzzBabe accuracy %, averaged over all training set sizes, for each experimental condition.

It can be seen from Table 27 that of the six conditions, 1, 3 and 5 show a progressive improvement in the relative performance of NonfuzzBabe, CorrBabe and FuzzBabe respectively. Conditions 2 and 4 show an improvement in fuzzy vs. non fuzzy performance but little difference between degrees of fuzziness. In condition 6, non fuzzy performs best, however, there is a positive correlation between the accuracy of fuzzy versions and degree of fuzziness in each implementation.

It is also evident from Table 27 that verb-only conditions show significantly degraded performance over sentential tests (by some 18 percent overall). This phenomenon is a consequence of the nature of the Baby model. With Features derived from sentential constructs of similar type, the ratio of the number of characters that form Constants within the FeaturePart of a Feature and that FeaturePart's length, is generally greater than the same ratio measured with FeatureParts derived from verb-only representations. Consequently, during the pattern extraction process, small, chance character commonalities within verbs become insignificant compared with larger sentential commonalities that generally include the verb inflection. Then during conversationally interactive phase, *COTLPin* enables Baby to detect Features that correspond to utterance structure that includes the verb inflection. This process has the general effect of increasing the signal to noise ratio between the verb

inflection and minor commonalities within the verb base. It is not surprising, therefore, that verb-only performance is generally poor, and in particular, that in the most difficult of the conditions (predicting unseen verbs), verb-only results provide the poorest performance. This phenomenon was detected early in the research project and formed part of the strategy underlying the decision to measure sentential constructs as well as verb-only.

7.5.3 Comparative test with other research models.

The data discussed and analysed in this section are selected to establish an indication of how the Baby model, when applied to past tense formation, compares with other researchers' implementations using different techniques to achieve similar tasks. Baby research is still at an early phase, consequently there are many parameters employed by Baby models that, if changed, may well significantly improve its performance. It was not expected that Baby would outperform other well known, tried and much researched acquisition engines. However, it was thought prudent to provide comparative data to indicate a reference. Implementational details of each of the other works lay outside the scope of this thesis. Each of the other models acquires past tense behaviour by exposure to verb-only training and test sets, whilst data relating to Baby models includes both verb-only and sentential training and test sets. This is because Baby models exhibit improved performance with sentential representations and, as stated earlier, the aim is to compare past tense acquisition performance and not the relative benefit of different input/output representations.

7.5.3.1 Rumelhart and McClelland models.

The Rumelhart and McClelland (1986) model is implemented as a simple perceptron-based pattern associator, connected to an input/output coding/decoding network. Verbs are input and output in the form of Wickelphone/Wickelfeature phonetic representations. Direct comparison between various implementations of Baby and the Rumelhart and McClelland model is difficult. This is because the tests applied by Rumelhart and McClelland to their model use a different and smaller corpus from which to extract their training and test sets. Also Rumelhart and McClelland designed their experiment to test for the psychological plausibility of their model. For instance, they divided their training and test sets into high, medium and low frequency verbs. Also they distinguish, not only between regular and irregular verbs, but sub divided these classes into nine types of irregulars with different inflection patterns and three types of regulars with different phonetic patterns.

However, Ling (1994) summarises the Rumelhart and McClelland model for error rates in both unseen regular and unseen irregular prediction. Though Ling does not say to what training corpus size his summary refers, Rumelhart and McClelland (1986) reveal that 420 verbs were used in their

experiment. Consequently, Baby data refers to the closest training set size of 500 verbs. Table 23 shows unseen, regular verb percentage prediction accuracy for the Rumelhart and McClelland model and average comparative data for NonfuzzBabe, CorrBabe and FuzzBabe.

It can be seen that Rumelhart and McClelland's model generally outperforms all Baby implementations, however comparison with sentential tests using FuzzBabe show the best performance. It should be noted that the Rumelhart and McClelland model was designed specifically for acquiring past tense behaviour, unlike the Baby model that has been shown to be capable of processing a wider domain of language. A general feature of connectionist learning that is shared by Rumelhart and McClelland's model, is that a positive correlation exists between accuracy and training set size. Section *7.2 Results of comparative tests using larger training and test corpora.*, shows that this is not necessarily the case with Baby implementations. Consequently, if a comparison is made between the best performance of each model, a more encouraging result emerges. Table 28 illustrates this phenomenon.

	McRum	NonfuzzBabe	CorrBabe	FuzzBabe
Verb-only	66	58	42	42
Sentential structure	N/A	57	75	78

Table 28: Comparison between the best performance of Rumelhart & McClelland models and NonfuzzBabe, CorrBabe and FuzzBabe.

7.5.3.2 Ling model.

Rather than choosing a selection of other researchers' work with which to compare Baby models, experiments conducted by Charles Ling of the University of Western Ontario on past tense learning provided a single source of useful comparative data. Following Rumelhart and McClelland; MacWhinney and Leinbach (1991) developed an improved connectionist model that was claimed by them to be superior to other past tense acquisition models thus far implemented. They were apparently so confident in their model that they issued a challenge in the same paper by asking the question "Is there a better symbolic model?". Ling (1994) rose to the challenge by producing four models, including both connectionist and symbolic that learned past tense behaviour. He presents results that show one of the symbolic models to have better performance than even the best connectionist models. Consequently, Ling's paper arguably contains data showing the best performing connectionist and symbolic models for the purpose of acquiring past tense behaviour. No other published work has been found that is dated since 1994 that outperforms Ling's results, within comparatively similar experimental test conditions.

Ling, (1994) describes four different past tense acquisition engines, two employ subsymbolic and two employ symbolic implementation. Details of the implementations are outside the scope of this thesis. In his paper, Ling provides training accuracy data and data relating to the accuracy of predicting unseen verbs for each of his models. The verbs used in each of Ling's models and also each Baby model are extracted from different random selections of the set of regular past tense formations in the MacWhinney corpus. Ling used one run in his tests whilst Baby data show average performance of three runs. Also, Ling uses tests set of various sizes whereas Baby models use test sets containing 30 different verbs.

Table 24 and Table 25 show overall disappointing Baby model performance, though this was, as argued previously, an expected outcome. The best that can be said is that Baby models outperform all other models when exposed to a training set size of 50 (using sentential representations), and outperforms both connectionist models even when exposed to a training set comprising 100 verbs. However, as training set size increase, so all other models exhibit far better performance than Baby models. Because both Ling and Baby models show peak performance with different sized training sets, it was decided to present an analysis here that shows the performance of Ling and Baby models averaged over all training set sizes. Table 29 shows this data for verb-only and sentential tests applied to NonfuzzBabe, CorrBabe and FuzzBabe and verb-only tests applied to Ling's models. SPA is used here as an abbreviation for Symbolic Pattern Associator.

	SPA 1	SPA 2	ANN 1	ANN 2	NonfuzzBabe	CorrBabe	FuzzBabe
Verb-only	79.8	70.4	56.2	48.6	42.8	29.6	32.6
Sentential					56.4	54.2	59.2

Table 29: Comparison between prediction accuracy, average over all training set sizes, of Ling models and NonfuzzBabe, CorrBabe and FuzzBabe.

It can be seen from Table 29 that in verb-only tests, no Baby model (with the possible exception of NonfuzzBabe) approaches the performance of any of Ling's models. However, when comparing verb-only with sentential, Baby models equal or exceed ANN models but perform less well than either SPA model.

7.5.4 Illustration of an emergent behaviour.

During research into Baby, many interesting behaviours were noted. Two have been chosen for inclusion in this thesis. One is presented here and the second, that of a correspondence between human and Baby early acquisition, is presented in the next chapter. The results presented in section *7.4 Results of processing text with no white-space.* and discussed here are intended to illustrate an interesting aspect of Baby behaviour rather than as part of a properly designed experiment.

7.5.4.1 Processing textual representations of language containing no white-space.

Segmentation is an interesting aspect of language processing. Textual representations of language usually comprise letters interspersed with white-space characters to separate words. However, the concept of 'word' is difficult to define (cf. Greenberg, 1963; Lehman, 1962). Elman (1990) demonstrates a recurrent network that is able to detect word boundaries in textual representations of language, from which white-space has been removed. Many symbolic NLP systems that model a human theory of the nature of language, rely upon white-space to detect word boundaries. This is required so that words may be symbolised (for instance by syntactic category) so that utterance structure may be parsed by reference to a lexicon and re-write rule base. Apparently, human speech processing does not require the white-space equivalent, the pause, between words. In normal speech, the interval between words is often absent, e.g. Markowitz, 1996.

It is recognised that humans find the comprehension of text containing concatenated letters very hard. For Baby implementations, the use of textual representation was a self imposed restriction in order to simplify the input set. In principle the Baby concept could be externalised to process many other representations including morphemes, phonemes, diphones speech waveform data for example. Consequently, it was thought legitimate to include an illustration of the ability of the Baby model to process concatenated letter processing here. Table 26 shows pluralisation behaviour using concatenated letter training and test corpora. It is, in principle, irrelevant to the Baby model whether word delineation is included or not in training and test sets, provided they are consistent. This is because the notion of 'word' does not form part of Baby theory, consequently NonfuzzBabe, CorrBabe and FuzzBabe will perform equivalently with or without white-space. FuzzBabe was used in this test because it processes pluralisation behaviour without error and this makes human reading of the test material less difficult.

7.6 Summary.

Results have been presented, discussed and analysed concerning a series of experiments performed upon NonfuzzBabe, CorrBabe and FuzzBabe. The experiments were designed to test the second hypothesis connected with this research programme. Firstly, it was shown that with small, contrived training and test sets, a fuzzy formalism, externalised in Baby's classification algorithm, did not suffer from inaccuracy and over-productivity that can be a problem with this formalism. With these tests, CorrBabe and FuzzBabe, though performing equivalently, on average outperformed NonfuzzBabe by a factor of some 27%.

Encouraged by this outcome, an experiment was designed and conducted, to test whether similar results could be observed when NonfuzzBabe, CorrBabe and FuzzBabe were applied to larger, non-contrived and thus more representative training and test sets. The domain of past tense formation was chosen as the language behaviour. Three Baby parameters were tested, training accuracy, prediction accuracy for seen verbs and prediction accuracy for unseen verbs, and each parameter was tested with verb-only and sentential representations of past tense behaviour. Results of these experiments were complex, however, average results for all test set sizes on each of the six conditions showed that NonfuzzBabe, CorrBabe and FuzzBabe processed sentential representations consistently better than verb-only representations. This phenomenon was explained. Furthermore, when testing with sentential representation, there was a positive correlation between accuracy and increasing fuzzy classification, whereas when testing with verb-only representation, this correlation was either small or absent.

Having shown some measure of the relative performance between NonfuzzBabe, CorrBabe and FuzzBabe, the results were used to compare with other researchers' work in the same field. Research that represented best performance for predicting unseen regular verbs using verb-only training and test sets that employed both symbolic and subsymbolic processing was chosen and the results presented. It was argued that, though not directly comparable, both sentential and verb-only test represented past tense learning behaviour and thus both would be presented for Baby models. In line with expectation, and for all implementations of Baby, comparative performance using verb-only training and test sets was poor. However, when comparing implementations of Baby using sentential representation with other models using verb-only representation, Baby results slightly exceeded ANN performance but remained worse than SPA performance. Bearing in mind that Ling's work represented best performance, and that Baby is in the early stages of development, these results were considered encouraging.

Finally, some results were presented that illustrated the ability of the Baby model to process language that contained no white-space.

8.0 Illustrations of advanced behaviour.

So far in this thesis, issues concerned with the various Baby models have been dealt with in a rigorous manner. In this chapter, advanced Baby behaviours are addressed. This is done by presenting them for illustration purposes only, consequently their treatment is conducted with less rigour than in previous research described here. These issues should be investigated more fully, but that effort lies outside the domain of the research described in this thesis. The chapter begins with a review of the kind of interactional discourse that the Baby models presented so far, would not be able to process. This is followed by a brief account of one method of how Baby may be augmented to cope with such interaction. A worked example is then given that illustrates the mechanism of an augmented Baby model that is processing an advanced behaviour. There follows a short report of some preliminary work carried out to explore this behaviour further. Results are then shown that indicate an interesting correspondence between an augmented Baby model, and aspects of the manner in which children acquire language during early stages. Finally, a summary is given of the issues covered in this chapter.

8.1 Natural language that Baby models presented so far cannot process.

Though the performance of NonfuzzBabe, CorrBabe and in particular FuzzBabe was found to be encouraging, such models cannot process language that contains many of the complexities described in section *2.4.2 Domain size vs. Processing* methods. Inaccuracies confuse processing and incompleteness, imprecision and ambiguity can only be resolved if Baby models were able to simulate an understanding of a world outside that contained within the stimulus/response elements of individual TextCouplets that produced the target FeatureCouplet for a novel utterance. Many of the complexities of language are concerned with ambiguity. The preliminary research described here is concerned with models of Baby that may be able to process language exhibiting this phenomenon.

8.1.1 Linguistic ambiguity.

The term ambiguity is often misused to mean semantic ambiguity, (multi-meaning at word level). Linguistic ambiguity subsumes semantic ambiguity and its ontology bounds all linguistic uncertainty. Ambiguity is an ubiquitous language phenomenon and consequently, many NLP systems have been developed to resolve it. Woods (1978), Schubert and Pelletier (1982), Alshawi (1992), Hobbs, Croft, Davies, Edwards & Laws (1987), or Grosz, Appelt, Martin & Pereira (1987), example toy applications developed during the theory-laden era. More recent work that processes language employing statistical analysis are exampled by Giachin, Lee, Rabiner, Rosenberg, & Pieraccini (1992), Ney, Essen & Kneser (1994a), Cutting, Kupiec, Pedersen & Sibun (1992),

Armstrong, Russell, Petitpierre & Robert (1995), Bod (1992) or Sima'an (1996). Aspects of ambiguity that may be considered for assessing the performance of a suitably augmented Baby model include, but are not confined to (compiled from Allen 1995):

- **quantifier scoping:**

e.g. "A dog entered with every man."

meaning: there are many dogs each entering with a new man?
a single dog enters with all the men together?
a single dog enters many times, each time with a new man?

- **anaphora (both in its local and global form):**

e.g. "Jane had a shower, she felt very refreshed."
to what does *she* refer?

- **ellipsis:**

e.g. "Jack forgot his wallet and Sam did too."
to what does *too* refer?

- **word sense:**

e.g. "Birds like fruit flies."
birds with the attributes of fruit, fly in the air. vs. birds are partial to fruit fly insects.

- **deduction, abduction and inference:**

e.g. to abduce that the traffic is congested from the fact that John is late to work.

- **intersectivity of adjectival phrases:**

e.g. is a slow tiger as slow as a slow snail?

- **comparatives:**

i.e. the degree of modification afforded by adjectival constructs;

e.g. how happy is Sue given that she is happier than Jane?

- **generics:**

e.g. "Lions are dangerous." is true even if some lions are tame, or
"Sea turtles lay about 100 eggs" is true even though most turtles are male.

- **prepositional attachment:**

e.g. "John killed the man with a knife."
was it John, or the man he murdered, that possessed a knife?

An augmented Baby model must simulate at least some of these aspects of human language behaviour, if it is to achieve any real degree of success. The encoding/decoding of a discourse from/into human concepts, cannot be simulated if the utterances comprising the discourse are considered to be discrete and unrelated, e.g. McKevitt (1992) or Coulthard (1992). Instead, elements of discourse comprise sets of complex, dynamic, interactional entities of syntax, meaning and topic

that are processed with relation to the language user's understanding of their environment. Perhaps a more convenient way to view these issues, is from the perspective of the knowledge that a user has of their language, and their ability to use that language appropriately. These abilities will be referred from hereafter as 'language knowledge' and 'appropriacy' respectively. It is postulated that the kind of language interaction that has been shown in experiments with NonfuzzBabe, CorrBabe and FuzzBabe can be considered to be examples of language knowledge. Further, that the kind of language behaviour required of augmented version of Baby models, is that which demonstrates both language knowledge and appropriacy.

8.2 FuzzyBaby.

This section provides a brief description of the rationale behind, and the basic principles involved with the design, implementation and performance indication, of a model of Baby that may exhibit some aspects of both language knowledge and appropriacy. The name coined for this implementation was FuzzyBaby.

8.2.1 Rationale.

Baby models presented so far in this thesis are incapable of resolving language containing many types of uncertainty. This is because the information required to disambiguate language containing these phenomena may lie outside the TextCouplets that produced the target FeatureCouplet for a novel utterance. Baby models must therefore be augmented to encompass wider domains within their Feature and Meta-Feature databases. This issue may be viewed from the perspective of the Baby's underpinning theory. It would seem that it is not sufficient for an hypothesised response to a new stimulus to be constructed purely upon the grounds of *correspondence*. This is because correspondence does not take *context* into account. In view of the results derived from research into Baby models presented so far, it is hypothesised that the underpinning theory may have been insufficiently explanative in this respect. For a theory of NLP to encompass language knowledge as well as appropriacy, it is necessary to account for context as well as correspondence. Consequently, for the purpose of researching advanced behaviour, the underpinning theory is augmented as follows:

for a given user of a language, a textual representation of an element of the response state engendered by experiencing a current textual stimulus, can be synthesised from textual response patterns associated with previously experienced stimuli, whose textual patterns most closely correspond with, ***and are contextually most similar to***, the textual patterns of the current textual stimulus.

This augmentation may be modelled in externalisation of the theory, by considering the concept represented by the words "most closely correspond" to relate to fuzzy classification of Constant patterns, and the concept represented by the words "contextually most similar" to relate to fuzzy classification of Variable elements.

Using the concepts of language knowledge and appropriacy, an experimental hypothesis can be formulated:

> Baby models process language knowledge from information contained in the FeaturePart of Features, and appropriacy from information contained in the VariablePart of Features.

Experiments described later in this chapter seek to test this hypothesis.

8.2.2 Principle of operation.

FuzzyBaby is designed to provide data that tests this hypothesis. It is identical in all respects to FuzzBabe other than that one extra fuzzy indicator is used in the classification process.

8.2.2.1 Definitions.

In addition to the definitions set out previously in this thesis, there follows a list of definitions for terms that describe concepts that are invoked for FuzzyBaby as follows:

1. **ProtoVariable:** a **VariableCluster** that is derived by removing the **SignaturePart** of a **Feature** from the input utterance. In the case where only an element of a **SignaturePart** is possessed by the input utterance, then the **VariableCluster** that would have been derived had the whole **SignaturePart** been present in the input utterance, forms the **ProtoVariable.**

   e.g. the **ProtoVariable: //cricket b//** is derived from:

   "Where's your cricket ball?" and **]0[Where is your]1[at?]0[**

2. **VariableCorrespondence ratio:**

   Given a **ProtoVariable** ***A$*** and a **VariableCluster** ***B$*** then:

   In the case where a concatenated element, that shares the two left-most characters of at least one Variable of ***B$***, exists in the same order, left to right, in ***A$*** then the **VariableCorrespondence ratio** is defined as:

$$\frac{\text{the length of common sub-Strings between } A\$ \text{ and } B\$}{\text{the length of } A\$}$$

e.g. given an input utterance of : **"Who's hot"**

and the **FeaturePart** of a **Feature:** **"][s] [.]["**

the **ProtoVariable** generated is: **/Who'/hot/**

if the **Feature's VariablePart** contains: **~/Jane/hot//~**

then the **VariableCorrespondence ratio** $= \frac{\text{length of "hot"}}{\text{length of "Who'hot"}} = \frac{3}{7} = 0.42857$

8.2.2.2 Brief description of the FuzzBabe mechanism.

FuzzyBaby operates in the same way as FuzzBabe up to, and including, the stage of generating the SecondaryCrispList that holds the set of SecondaryActivatedCouplets that are generated by FeatureCouplets that possess the largest FuzzyCorrespondence ratio with the input utterance, i.e.:

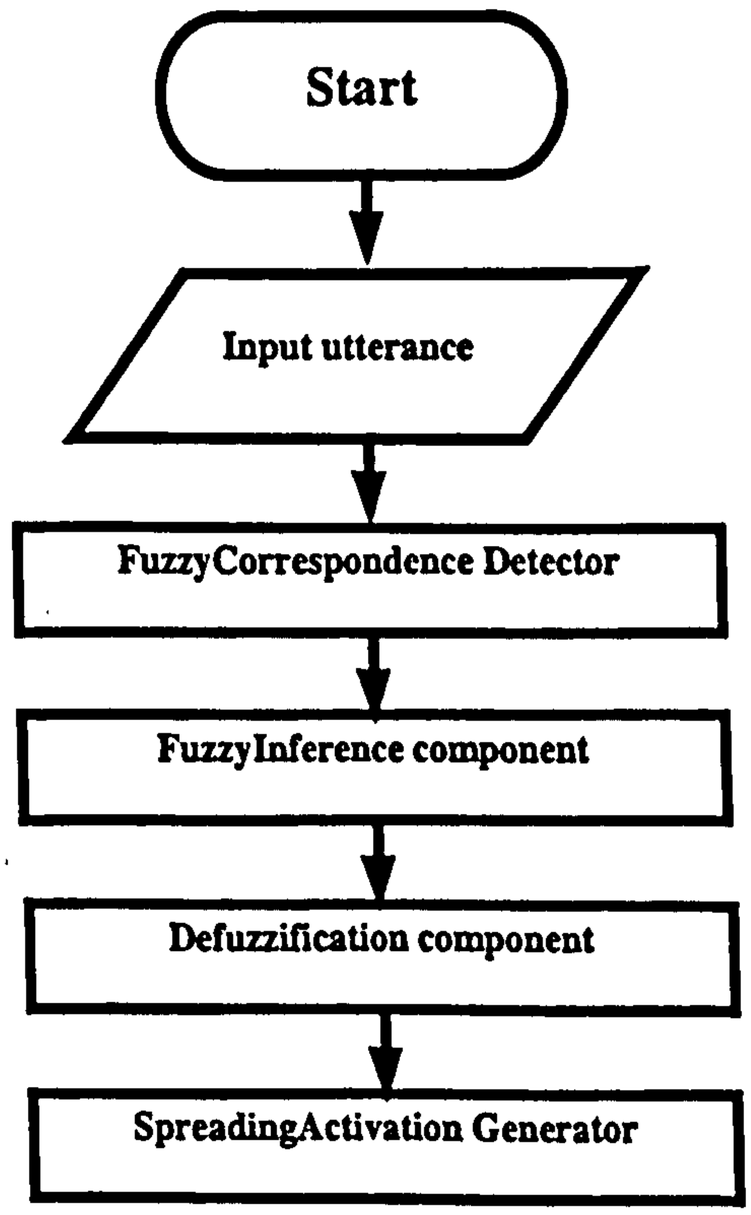


Figure 21: Flow diagram of FuzzBabe's Pattern Comparator module (up to the output of the SpreadingActivation generator).

In addition to this process, FuzzyBaby takes the PrimaryCrispList and the SecondaryCrispList output in process shown in

Figure 21, and evaluates a set of fuzzy variable indicators as follows. For each Feature in the PrimaryCrispList, a ProtoVariable is generated between the FeaturePart of a Feature and the input utterance. Then for each Feature in the SecondaryCrispList, VariableCorrespondence ratio is evaluated between the ProtoVariable and all the VariableClusters contained in the VariablePart of the Feature. If VariableCorrespondence ratio(s) exit, then the address(es) of the FeatureCouplet(s) along with their VariableCorrespondence ratio(s) are stored in the FuzzyVariableList. This list is passed to a defuzzification component that performs max-membership principle evaluation on the VariableCorrespondence ratio(s). This component produces a DefuzzifiedVariableList containing the address(es) of Features along with their VariableCorrespondence ratio(s). The DefuzzifiedVariableList is added to the SecondaryCrispList to form duplicate addresses. If instances of VariableCorrespondence are found, then their multi-occurrence in the SecondaryCrispList, results in a greater chance of them becoming the WinningResponse. This is another manifestation of *COTLPin*. Figure 22 shows a process flow diagram of these modifications for the Pattern Comparator module of FuzzyBaby

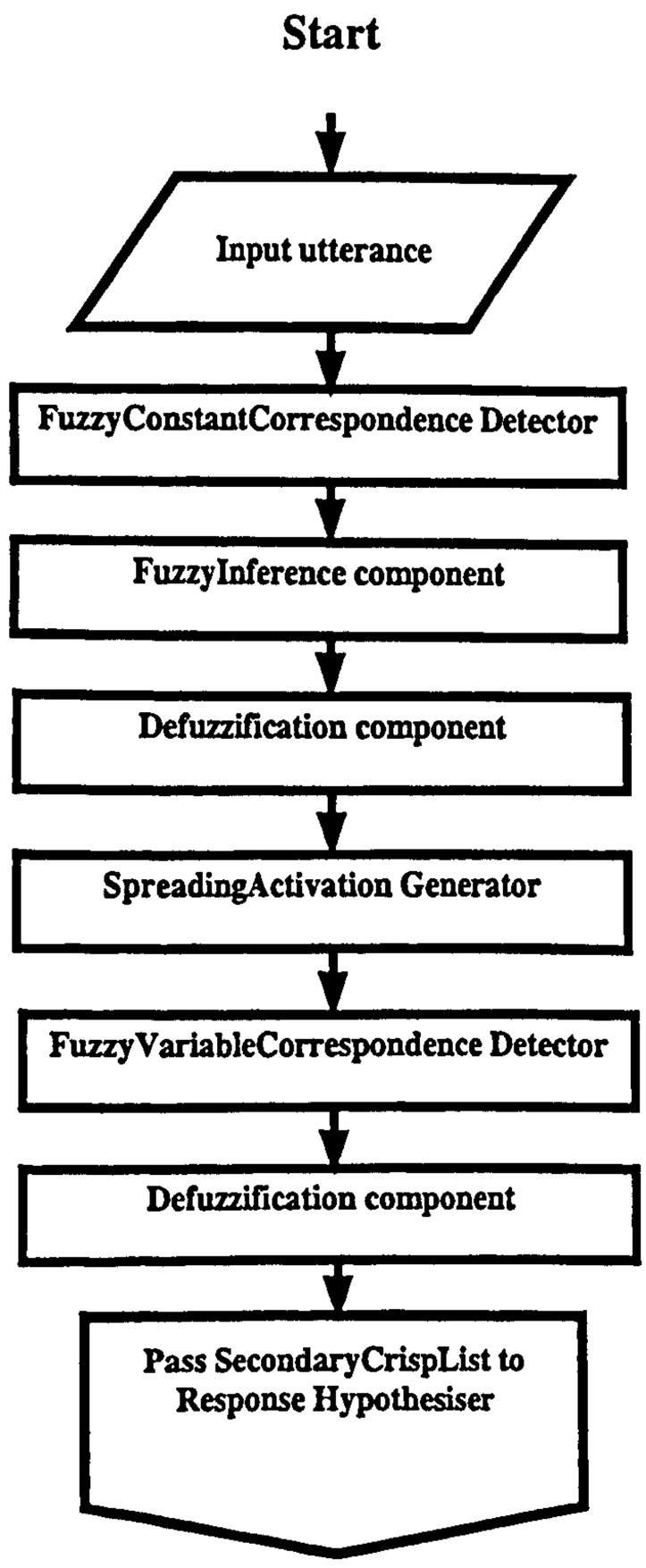


Figure 22: Flow diagram for Pattern Comparator module for FuzzyBaby.

8.2.3 A worked example.

The following text and tables show the detail of fuzzy Constant and fuzzy Variable processing that result in a FuzzyBaby model processing the same correct response, for three different ambiguous conversational stimuli, from the same FeatureCouplet/MetaCouplet database, via different routes. Each conversational stimulus has the same sense, but stimuli 2 and 3 are presented with successively less textual information linking in them to their target Feature. It should be noted, that the training set does not include a conversational pair that represents a query/answer structure that unambiguously determines the correct response to the input utterance. Note also the use of the pronominal reference "who". Table 30 shows the three conversational stimuli along with FuzzyBaby's response in each case.

No.	Human conversational input	FuzzyBaby's conversational response
1	Who is hot?	Jane is hot.
2	Who is hot	Jane is hot.
3	Who's hot	Jane is hot.

Table 30: Input output utterance used to illustrate FuzzyBaby's performance.

There is no special significance to the stimulus 'Who is hot'; it is employed simply to illustrate complex behaviour.

There are two training sets used, a query set comprising examples of different types of query/response structures, and a pronominal training set comprising examples of statement stimuli each followed by an utterance that include a pronominal reference.

Query training set

Table 31 shows the query training set. It comprises two examples of each of the following query structures: ***What is a X?***; ***Who is X?; Why is X a Y?; Is a X a Y?*** and ***Where is the X?***. For the training set, textual values for *X* and *Y* were chosen randomly, but care was taken to avoid choosing values that shared Constants. This is because with such a small training set, FuzzyBaby processes Constants contained in the textual values for *X* and *Y*, as typical of the structure of the query type rather than chance occurrences.

What is a man? **A man is a beast.**	**Why is Linda a knowall?** **Because Linda is a genius.**
What is a carrot? **A carrot is a vegetable.**	**Is a goat a friend?** **A goat is a friend.**
Who is Jane? **Jane is a girl.**	**Is a pig a sheep?** **A pig is a sheep.**
Who is Peter? **Peter is a boy.**	**Where is the plate?** **The plate is in the kitchen.**
Why is John a twit? **Because John is a dimo.**	**Where is the car?** **The car is in the garage.**

Table 31: Query training set.

Pronominal training set

Table 32 shows a training set comprising examples of statement stimuli followed by utterances that include a pronominal reference. For the same reason as with the query training set, care was taken to avoid extraneous Constant formation.

Jane is hot. **It is a delight.**
Peter is dozing. **He is a sleeper.**
Potato is lovely. **Colin eats them.**
Running is exercise. **Paul does it.**

Table 32: Pronominal training set.

The Pattern Extraction process may operate on each training set in turn, or both together, or indeed TextCouplets in both training sets may be mixed randomly. Table 33 shows FeatureCouplets each with their VariableParts, that are generated by this process. MetaFeatures exist and are used in processing, (see Table 34). The leftmost column in the lists of Features and MetaFeatures is for numerical identification purposes:

PAGE NUMBERING AS ORIGINAL

FeatureCouplets

1.	]0[Where is the]1[?]0[	//plate//~//car//~
2	]0[The]1[is in the]2[	//plate/kitchen./~//car/garage. /~
3	]0[What is a]1[?]0[	//man//~//carrot//~
4	]0[A]1[is a]2[.]0[	//man/beast//~//carrot/vegitable//~
5	]0[Who is]1[?]0[	//June//~//Peter//~
6	]1[is a]2[.]0[	/June/girl//~/Peter/boy//~
7	]0[Why is]1[a]2[?]0[	//John/twit//~//Linda/knowall//~
8	]0[Because]1[is a]3[.]0[	//John/dimo//~//Linda/genius//~
9	]0[Is a]1[a]2[?]0[	//goat/friend//~//pig/sheep//~
10	]0[A]1[is a]2[.]0[	//goat/friend//~//pig/sheep//~
11	]0[is]1[.]2[	/Jane/hot//~/Peter/dozing//~
12	]3[is a]4[.]2[	/It/delight//~/He/sleeper//~
13	]0[is]1[.]2[	/Jane/hot//~/Peter/dozing//~/Potato/lovely//~/Running/exersise//~
14	]3[s]4[.]2[	/It i/a delight//~/He i/a sleeper//~/Colin eat/them//~/Paul doe/it//~

Table 33: FeatureCouplets with their VariableParts computed with query and pronominal training sets.

Meta(1)Features

15	]0[Wh]1[is]2[?]0[	//at/a man//~//at/a carrot//~//o/June//~//o/Peter//~//ere/the plate//~//ere/the car//~
16	]2[is]3[	/A man/a beast./~/A carrot/a vegitable./~/June/a girl./~/Peter/a boy./~/The plate/in the kitchen./~/The car/in the garage. /~
17	]0[Wh]1[is]2[?]0[	//y/John a twit//~//y/Linda a knowall//~
18	]3[is]4[	/Because John/a dimo./~/Because Linda/a genius./~
19	]0[Is]1[a]2[?]3[	/Why /John/twit//~/Why /Linda/knowall//~
20	]4[]5[is a]6[.]3[	/Because /John/dimo//~/Because /Linda/genius//~
21	]0[Is]1[a]2[?]0[	//a goat/friend//~//a pig/sheep//~
22	]1[is a]2[.]0[	/A goat/friend//~/A pig/sheep//~
23	]0[is]1[.]2[	/Jane/hot//~/Peter/dozing//~/Potato/lovely//~/Running/exersise//~
24	]3[s]4[.	/It i/a delight//~/He i/a sleeper//~/Colin eat/them//~/Paul doe/it//~

Table 34: MetaCouplets with their VariableParts computed with query and pronominal training sets.

The procedure so far, is common for all three input utterances. Table 35 shows a set of data that maps the process of fuzzy Constant and fuzzy Variable information that enables FuzzyBaby to process the responses to the stimuli shown in Table 30. In descending order, the rows in Table 35 show the following data:

- the contents of the PrimaryCrispList following FuzzyCorrespondence processing;
- the contents of the SecondaryCrispList following FuzzyCorrespondence processing ;
- the FuzzyCorrespondence ratio for each of the Features in the SecondaryCrispList;
- ProtoVariables formed between input utterance and Features in the SecondaryCrispList;
- the address of Features that share a VariableCorrespondence ratio with input utterance;
- the address of Features whose VariableParts possess a VariableCorrespondence ratio;
- the VariableCorrespondence ratio of those Features;
- the list of CompetitiveResponses generated.
- the list of WinningResponses.

Input utterance	Who is hot?	Who is hot	Who's hot
PrimaryCrispList (constant correspondence)	5	5	14 24
SecondaryCrispList (following the spreading activation process)	6 12 22 11 13 23 19 21	6 12 22 11 13 23 19 21	13 23 11 6 12 22
FuzzyCorrespondence ratio (constant correspondence)	1 1 1 1 1 1 1 1	0.7656 0.7656 0.7656 0.7656 0.7656 0.7656	0.1975 0.1975 0.1975 0.1975 0.1975
ProtoVariables	//hot// //hot// //hot// //hot// //hot// //hot// //hot// //hot//	//hot// //hot// //hot// //hot// //hot// //hot// //hot// //hot//	/Who'/hot/ /Who'/hot/ /Who'/hot/ /Who'/hot/ /Who'/hot/ /Who'/hot/
SecondaryCrispList (variable correspondence only)	11 13 23	11 13 23	11 13 23
Activated VariableClusters	/Jane/hot// /Jane/hot// /Jane/hot//	/Jane/hot// /Jane/hot// /Jane/hot//	/Jane/hot// /Jane/hot// /Jane/hot//
VariableCorrespondence ratio	1 1 1 1	1 1 1 1	0.4285 0.4285 0.4285
CompetitiveResponses	hot is a (...). (...) is a (...).(...) hot is a (...). (...) is a (...).(...) (...)s (...).(...) (...)s (...).(...) (...)(...) is a (...)(...) hot is a (...). Jane is hot. Jane is hot. Jane is hot. How should I reply?	hot is a (...). (...) is a (...).(...) hot is a (...). (...) is a (...).(...) (...)s (...).(...) (...)s (...).(...) (...)(...) is a (...)(...) hot is a (...). Jane is hot. Jane is hot. Jane is hot. How should I reply?	(...)s (...).(...) (...)s (...).(...) (...) is a (...).(...) hot is a (...).Who' (...) is a (...).(...) hot is a (...).Who' Jane is hot. Jane is hot. Jane is hot. How should I reply?
WinningResponse	Jane is hot.	Jane is hot.	Jane is hot.

Table 35: Set of data that maps the process of fuzzy Constant and fuzzy Variable information generated in the worked example.

8.2.4 Brief analysis.

The first utterance produces an unambiguous response, as it shares ConstantEquivalence with a FeaturePart of a Feature, and the VariablePart of that Feature contains the complete ProtoVariable. This results in a FuzzyCorrespondence and FuzzyVariableCorrespondence ratio of 1. The second utterance shares partial correspondence with the same FeaturePart of a Feature as did the first utterance, but as the ProtoVariable is derived by removing the complete FeaturePart from the input

utterance, a unity FuzzyVariableCorrespondence is still evaluated. The third utterance is so dissimilar from the FeaturePart of the Feature that was activated by input utterances 1 and 2, that it shares a larger FuzzyCorrespondence with the FeaturePart of two different Features. Also a different ProtoVariable is formed with these FeatureParts. Consequently, both FuzzyCorrespondence and FuzzyVariableCorrespondence ratios are different than with input utterances 1 and 2. However, such is the nature of the FeatureCouplet/MetaCouplet database generated by the Pattern Extraction process on representative training sets, that FuzzyBaby processes the same response as for utterances 1 and 2. This response is considered correct within the context of the information available in the training sets.

Although NonfuzzBabe, CorrBabe and FuzzBabe form no part of the rationale for this study, within the context of this thesis, their performance when applied to the worked example is significant. Table 36 shows these results.

	Who is hot?	**Who is hot**	**Who's hot**
NonfuzzBabe	o is hot.	How should I reply?	How should I reply?
CorrBabe	How should I reply?	How should I reply?	How should I reply?
FuzzBabe	How should I reply?	How should I reply?	How should I reply?

Table 36: Results of the application of the worked example on NonfuzzBabe, CorrBabe and FuzzBabe.

8.3 Some preliminary research that investigates advanced Baby behaviour.

In this section, preliminary results of a pilot study are shown. The main experiment has not yet been conducted. The rationale for the main experiment was to investigate the language domain of FuzzyBaby further. It was decided to apply FuzzyBaby to a test that is designed to assess young children's reading comprehension. The test chosen was one published by NFER-Nelson Publishing Company Ltd. The test is used extensively by schools in England. Despite repeated attempts by the author and Staff at Bournemouth University to gain permission to use the test material in this document, no such permission has yet been granted. Consequently, it is not possible to either reproduce parts of the test, or use the NFER-Nelson method to evaluate reading comprehension metric in this thesis. This was not thought sufficient reason to abandon the pilot study, because it was hoped that permission would finally be granted. The reading comprehension test comprises the child reading a passage of text and then answering six questions. The questions are designed to enable an evaluation of the child's reading comprehension. The questions contain ambiguity, for

example, it is necessary to employ explicit and implicit inferencing to answer them correctly. The results of the pilot study have given confidence to design and perform a more extensive and better controlled experiment. If this transpires, it is intended to include them in a paper that will be submitted for journal publication.

8.3.1 The Ostrich test.

The comprehension test chosen was one whose reading text concerned the habits of Ostriches and their relationship with man during the past 2000 years. At their present stage of development, Baby models are only able to process conversationally interactive text. It is therefore not possible for them to read a passage of text and answer questions concerning meaning contained in that text. This is because the reading material does not contain language behaviour that examples the structure of the interaction concerned with answering the test questions. In order to overcome this difficulty, it was decided to ask adult participants to read the comprehension text. The participants were then asked to think of five questions that they might ask a child of about five years of age, concerning meaning contained in the text. Along with each question, participants were asked to provide answers that they would consider to represent a correct response. In the pilot study, six participant were used, each providing five query/response couplets. Twenty five of the couplets provided by them were used as a training corpus and the remaining five were used as a test corpus. Appendix C shows copies of these questionnaires. Table 37 shows the training corpus.

Where did the Ostriches live many years ago?
In Africa.

In Egypt who were allowed to wear Ostrich feathers and why?
Kings, because they were important.

Why did their feathers endanger the birds?
Because the hunters killed them so people could wear their feathers.

Why were there not many Ostriches left in Africa?
Because so many Ostriches were killed for fashion.

How do the farmers make the Ostriches lay more eggs?
They hide the eggs.

What kind of birds were worshipped by the people of Egypt?
Ostriches.

Who were the only people allowed to wear their feathers?
Kings.

Why were the Ostriches nearly made extinct?
Hunters killed many of them for their feathers.

How many eggs do they lay before they will sit on them?
Twelve.

How do the farmers get the birds to lay more eggs?
By hiding some of them.

Why did Kings wear Ostrich feathers?
Because of their beauty.

Why did the hunters kill so many Ostriches?
To sell the feathers for people to wear.

What was the result of so many Ostriches being killed?
They nearly became extinct.

How many eggs do Ostriches lay before sitting on them?
Twelve eggs.

How do Ostrich farmers make the Ostriches lay more eggs?
By hiding some of them.

Where do Ostriches come from?
Ostriches come from Africa.

Which people once worshipped Ostriches?
The people of Egypt once worshipped Ostriches.

Why were Ostriches hunted?
Ostriches were hunted for their feathers.

How many eggs do Ostriches lay?
Ostriches lay about twelve eggs.

How might you fool an Ostrich into laying eggs?
By hiding some that have already been laid.

What did you used to find in Africa?
Ostriches.

Why did people used to hunt Ostriches?
So they could sell or wear the feathers.

What was the result of killing so many Ostriches?
They nearly died out.

What do Ostriches lay?
Eggs.

How do you make Ostriches lay more eggs?
Hide some of them.

Table 37: Training corpus for the Ostrich test.

Table 38 shows the five participant query/answer couplets used to form the test corpus:

How do farmers make Ostriches lay more eggs? Hide some of them. Can you remember the name of the animal in the story? Ostriches. What country did Ostriches live in? Africa. Why did hunter kill Ostriches? Rich people bought the feathers. Where do Ostriches live today? Ostriches come from Africa.

Table 38: Participant query/answer couplets used to form the test corpus for the Ostrich test.

Table 39 shows the responses hypothesised by FuzzyBaby to the question elements in the test corpus following a Pattern Extraction process on the training corpora:

How do farmers make Ostriches lay more eggs? Hide some of them. Can you remember the name of the animal in the story? The people of Egypt once worshipped Ostriches. What country did Ostriches live in? How should I reply? Why did hunters kill Ostriches? Because of their beauty. Where do Ostriches live today? Ostriches come from Africa.

Table 39: FuzzyBaby's responses to the test corpus.

For ease of comparison, Table 40 shows the human response and FuzzyBaby response together:

No.	Human response	FuzzyBaby response
1	Hide some of them.	Hide some of them.
2	Ostriches.	The people of Egypt once worshipped Ostriches.
3	Africa.	How should I reply?
4	Rich people bought the feathers.	Because of their beauty.
5	Ostriches come from Africa.	Ostriches come from Africa.

Table 40: Human and FuzzyBaby responses to the test corpus.

It was tempting to provide an in-depth analysis of the results presented for the Ostrich test. However, it was decided not to pursue this course of action for two reasons:

- the experiment was intended as a pilot study and consequently, experimental design and control were minimal. Therefore, any inferences made from the data would not be sufficiently supported for inclusion in this thesis;
- inferences so far considered, but not printed here, have far reaching consequences in the areas of philosophy, cognitive science, and AI. Consequently a rigorous appraisal of them would be outside the scope of the research domain described in this thesis.

As a result, the data are presented here in order to support the design and execution of a properly conducted experiment, and also give an insight into the possible potential of the Baby concept to the discipline of NLP in particular and AI in general.

Although NonfuzzBabe, CorrBabe and FuzzBabe form no part of the rationale for this pilot study, within the context of this thesis, their performance when applied to the Ostrich test is significant. When exposed to the same test, each implementation hypothesises the "How should I reply?" string for all questions.

FuzzyBaby was also input with the test questions printed in the NFER-Nelson test. For the reason of copyright restriction, the text of the questions cannot be reprinted here. However, it is possible to report that of the six questions, FuzzyBaby responded to one question with ungrammatical nonsense, gave grammatically correct, but semantically incorrect answers to four questions, and answered one question correctly. This is a very hard test for FuzzyBaby because the syntactic structure of the questions was different than those chosen by the participants. Also, the NFER-Nelson questions required more world knowledge than those suggested by participants.

8.3.2 Correspondence with human language behaviour.

During investigation of Baby models, it was noted that their language behaviour resembled some aspects of human language behaviour. There follows a brief account of the second of two such behaviours shown in this thesis (see *7.5.4.1 Processing textual representations of language containing no white-space.*, for an account of the first behaviour). Illustrations of the two behaviours are separated in this manner, because the first is evident with NonfuzzBabe, CorrBabe, FuzzBabe and FuzzyBaby, whilst the second is manifest by FuzzyBaby only.

8.3.2.1 Language acquisition behaviour.

As a part of their language acquisition process, children acquire the ability to form the past tense of verbs in similar performance stages. Much psycholinguistic research has concentrated upon this aspect of language acquisition (e.g. Brown, 1973; Ervin ,1964; Kuczaj, 1977). The work suggests that children acquire the ability to form the past tense of verbs in a three stage process. In brief summary:

Stage 1. Small vocabulary of verbs in the past tense, predominantly irregular, e.g. take/took and come/came. No generalisation of the small set of regular past tenses that *are* known. Past tenses appear to be learned by rote rather than by rules.

Stage 2. Much larger vocabulary of regular past tense formations. Generalisation of regular inflections observed with new verbs and non-words. Generalisation overrides rote learning, causing previously known irregular past tenses to be incorrectly extended e.g. "taked" or "comed".

Stage 3. Greatly expanding vocabulary. Maintaining generalised inflection behaviour Re-acquiring correct exception past tensing. Formation of minor regular clusters, e.g. "ow" to "own", "ing to "ang")

Aspects of this behaviour are evident in the acquisition behaviour of FuzzyBaby. FuzzyBaby was exposed to a corpus comprising a set of conversationally interactive pairs, that represented the formation of the past tense of verbs (similar to those tests described in section *5.2 Test results.*). The verbs that were chosen, were divided into three types: irregular, regular and minor cluster regular. Each verb of each type was embedded to form three blocks of interactional utterances that formed the corpus as follows:

1. Six pairs embedded with irregular verbs;
2. Two pairs embedded with regular verbs;
3. Two pairs embedded with minor cluster regular verbs.

Interspersed between sections 1, 2 and following 3 were test queries that employed the same sentential structure as the first utterance of the utterance pairs used in each section.

Table 41 shows results of a single run test using formation of past tense constructs. The structure of the interactional pairs is as follows:

- utterances 1a through 6b represent the irregular section;
- utterances 7 and 8 form test queries;
- utterances 9a through 10b represent the regular section;
- utterances 11 and 12 form test queries;
- utterances 13a through 14b form the minor regular cluster section;
- utterances 15 through 19 form test queries.

For this test, the Pattern Extraction process was initiated every four conversational utterances (two human and two Baby) thus forming an interactive learning conversation (see chapter *4.0 NonfuzzBabe.*). It was for this reason that the various sections could be tested as exposure to the training/test corpus proceeded.

No.	Human utterance	FuzzyBaby's response
1a	What is the past tense of have?	How should I reply?
1b	The past tense of have is had.	OK
2a	What is the past tense of go?	How should I reply?
2b	The past tense of come is gone.	OK
3a	What is the past tense of give?	How should I reply?
3b	The past tense of give is gave.	OK
4a	What is the past tense of get?	How should I reply?
4b	The past tense of get is got.	OK
5a	What is the past tense of take?	How should I reply?
5b	The past tense of take is took.	OK
6a	What is the past tense of is?	How should I reply?
6b	The past tense of is is was.	OK
7	What is the past tense of give?	The past tense of give is gave.
8	What is he past tense of is?	The past tense of is is was.
9a	What is the past tense of train?	How should I reply?
9b	The past tense of train is trained.	OK
10a	What is the past tense of walk?	How should I reply?
10b	The past tense of walk is walked.	OK
11	What is the past tense of take?	The past tense of take is took.
12	What is the past tense of jump?	The past tense of jump is jumped.
13a	What is the past tense of sweep?	The past tense of sweep is sweeped?
13b	The past tense of sweep is swept.	OK
14a	What is the past tense of keep?	The past tense of keep is keeped?
14b	The past tense of keep is kept.	OK
15	What is the past tense of sleep?	The past tense of sleep is slept.
16	What is the past tense of part?	The past tense of part is parted?
17	What is the past tense of creep?	The past tense of creep is crept?
18	What is the past tense of have?	The past tense of gave is had.
19	What is the past tense of cleef?	The past tense of cleef is cleft.

Table 41: Results showing correspondence between human and FuzzyBaby language acquisition.

8.3.2.1.1 Brief analysis.

To compare the enormous exposure experienced during the early years of human language acquisition to the 19 interactions shown in Table 41 may be thought of as a rather biased test in favour of human language processing. However, partly as a consequence of the manner in which the test was contrived, a correspondence between FuzzyBaby's language learning behaviour and human acquisition is observable.

Stage 1. Initially, FuzzyBaby is exposed to a series of irregular past tense formations, exampled in utterances 1 – 6b. Lack of hypothesised response shows that no generalisation behaviour was taking place, though each past tense formation has been acquired by rote, as exampled in utterances 7 and 8. Utterances 9a and 9b represent the first example of a verb that possesses a regular past tense formation. At this stage FuzzyBaby has not been exposed to sufficient regular formation to have learnt the behaviour and therefore responds with the "How should I reply?" string.

Stage 2. Utterances 10a & 10b may be considered to represent a greater exposure to regular verbs learned by children during this stage. Utterance 11 shows that FuzzyBaby remembers previously learned exceptions and thus does not incorrectly hypothesise them with regular past tense inflections (unlike human performance). However, hypothesised response to new verbs are extended with the a regular inflection, e.g. "jump" and "jumped" in utterances 12. This behaviour extends through the first exposure to minor cluster regularities as shown in utterances 13a, 13b, 14a and 14b, producing the over-regularisation behaviour noticed in human language acquisition.

Stage 3. Utterance 15 confirms that the minor cluster regularity has been correctly learned. Utterances 16 and 17 show that FuzzyBaby is able to selectively extend root verbs in their past tense, whilst utterance 18 shows that irregular verbs are still correctly acquired. Utterance 19 shows the phenomena noticed by Prasada and Pinker (1993) where adults sometimes extend irregular inflections to irregular sounding regular or pseudo verbs.

8.4 Summary.

In this chapter, it was argued that the Baby models presented earlier in this thesis, though promising, could only process language in a limited domain. This domain subsumes language where the problem of generating a response to a conversational stimulus can be resolved by language knowledge. Though many FeatureCouplets may contribute to the list of competitive responses, the winning response must have been resolved by a single FeatureCouplet or a set of FeatureCouplets with identical FeatureParts. One of the consequences of this is that the Baby models so far

presented, cannot produce interaction that includes concepts introduced in TextCouplets of the training set that did not contribute to the FeatureCouplet that formed the winning response. An augmentation of the underpinning theory was postulated that may contribute to a resolution of this problem. It was hypothesised that externalisations that model the augmented theory would allow Baby models to exhibit appropriacy. A Baby model, FuzzyBaby, was briefly described that modelled the hypothesis. A worked example was then given that illustrated the mechanism of FuzzyBaby. A brief description, and the results obtained from a pilot study that further supported the hypothesis, were given. Finally, an illustration of a correspondence between human language acquisition behaviour and FuzzyBaby language acquisition behaviour was presented.

9.0 Summary, general discussion and conclusions.

This chapter begins with a summary of the research described in this thesis. It is followed by a discussion on issues that are better considered now that the research has been described in its entirety. Some conclusions that may be drawn from the research are then stated. The chapter ends with a comparison between the research aims and work done.

9.1 Summary.

There follows a summary of the research described in this thesis. The reader is referred to relevant chapters for more extensive explanations.

9.1.1 Introduction.

This research concerned a novel approach to AI machine learning. A general theory, developed by the author, that underpinned this approach may be seen in section *1.1 Context of research.* The theory reflected an acknowledgement that humans are good at solving complex problems and that cognition (particularly at low levels) could be thought of as a pattern association process. Consequently, if concrete patterns of behaviour were detected and stored from problem/solution sets contained in a domain, then these might be used to compute hypothetical solutions to unseen problems within that domain. Such an approach did not use rules that modelled the problem domain itself. Instead, instantiations of the underpinning theory used the problem domain to supply solution strategies directly from the complexity of the domain itself.

9.1.2 The test domain.

The test domain employed was the processing of textual representations of natural language interaction. The name Baby was adopted for this project because it was expected that the theory would be tested using externalisations that processed small sets of past experience. An adaptation of the general theory for the purpose of processing the test domain was:

for a given user of a language, a textual representation of an element of the response state engendered by experiencing a current textual stimulus, can be synthesised from textual response patterns associated with previously experienced stimuli, whose textual patterns most closely correspond with the textual patterns of the current textual stimulus.

Machines described in this thesis and which externalise this theory, detect patterns of characters that are coincident between examples of interactional discourse, then use these patterns to hypothesise responses to novel textual stimuli.

9.1.3 Rationale.

There were four main motivations behind pursuing this project:

- some intellectually challenging research entitled SEARS (see section *2.6.1 SEARS.*);
- the prospect that SEARS offered the possibility of overcoming some of the problems related to machines that acquire the behaviour of complex problem spaces;
- to test the underpinning theory (SEARS was seen as a vehicle to achieve this);
- NLP was recognised as being one of the most challenging problems within the AI discipline.

9.1.4 Related research.

Of the many disciplines associated with AI, the two that most closely related to this research were Data Oriented Language Processing and machine reasoning with uncertainty.

9.1.4.1 Data Orientated Language Processing.

DOLP was identified as that research area within NLP that most closely related to this work. A critical analysis was conducted between the two areas of research (see section *2.8.2 A comparison between Baby and DOP.*). It was discovered that the Baby concept lay outside main stream DOLP, but shared many of its underlying principles. No published work could be found that described research that copied the Baby approach.

9.1.4.2 Formalisms capable of reasoning with uncertainty.

In order to instantiate the underpinning theory, concepts represented by the words "most closely correspond" were required to be modelled. Modelling vague concepts required reasoning within nonmonotonic problem spaces. Consequently, it was necessary to research the more popular formalism capable of such reasoning. Possibility theory, manifest as fuzzy logic, was hypothesised as the most suitable contender for application with Baby models (see section *2.10.7 Analysis of formalisms capable of processing nonmonotonic problem spaces.*)

9.1.5 Theory of the Baby concept.

The Baby method of acquiring natural language was thought to be original, consequently it was necessary to create a new externalising model and to develop a set of concepts that underlies the principles behind the processes involved.

9.1.5.1 Modelling.

In order to build a machine that instantiated aspects of the underpinning theory, five attributes of neural processing were employed as model components:

1. attention;
2. forming hierarchical structures of patterns of data;
3. pattern recognition;
4. activation;
5. reasoning with uncertainty.

The number of character patterns that may be coincident between utterances is large, especially between those with similar structure. It was necessary to reduce the number considered, by concentrating upon a subset of all patterns. The five modelling components were used to produce a set of heuristics that characterised this subset. Characters that constituted patterns of the subset, would be:

- concatenated elements of the utterance;

where:

- larger elements took precedence over smaller elements;
- elements cannot exist within, or overlap, other elements;
- element length could not be less than two characters;
- larger elements prescribed the hierarchical structuring of smaller elements.

See section *2.7.4 Synthesis of model components.* for a worked example that illustrated the characteristics of the subset of patterns generated by the heuristics.

9.1.5.2 Implementation.

Simple externalisations of the Baby concept (coined NonfuzzBabe) (see section *4.0 NonfuzzBabe.*) comprised a feature extraction mechanism that took as input a training corpus, which simulated past language experience. The mechanism detected and stored textual patterns that complied with the

heuristics. Following this process, an equivalence classification mechanism took as input a novel utterance, compared it with the textual patterns processed previously, and produced a set of textual patterns that most closely correspond with input. Competitive, hypothetical response utterances were generated between input and the set of classified patterns. The most frequently occurring competitive, hypothetical response was chosen by Baby as its final hypothesised response.

Complex externalisations of the Baby concept (see section *6.0 Baby implementations employing fuzzy classification techniques.*) more accurately model correspondence between input utterance and textual patterns by incorporating fuzzy classification processes. Other than this, all aspects were identical to equivalence classification models.

9.1.6 Experimental design.

Two experimental hypotheses were developed that could be used to test such implementations. They were chosen in order to limit the test domain, to one that was both significant and commensurate with a research project of this size. The first hypothesis (see section *3.2.1 Hypothesis one*) was designed to test whether NonfuzzBabe was capable of processing natural language at any level of competence.

An implementation that incorporated equivalence pattern classification coined, NonfuzzBabe, was designed, implemented and tested. This implementation was used to test the first hypothesis.

The second hypothesis (see section *3.2.2 Hypothesis two*) was designed to test the effect of incorporating fuzziness in NonfuzzBabe's classification processes. This was decided upon because fuzzy formalisms may exhibit inaccuracy and/or overproductivity.

Two further implementations, CorrBabe and FuzzBabe were designed, implemented and tested, each employed progressively increasing fuzziness in their pattern classifier processes. All three implementations were employed to test the second experimental hypothesis.

9.1.7 Experimental tests.

One experiment comprising seven different language behaviours was performed to test the first hypothesis and performance indications were noted (see section *5.2 Test results.*). Two experiments were performed to test the second hypothesis. Firstly, both fuzzy implementations were applied to the same experiment used to test the first hypothesis and performance indications were noted (see section *7.1.3 Accuracy comparison between NonfuzzBabe, CorrBabe and FuzzBabe.*). This test was

not thought conclusive because of its limited nature. Consequently, a second experiment was designed to address this issue. In this second test, fuzzy implementations were applied to past tense acquisition using much larger training and test corpora and performance indications were noted (see section *7.2 Results of comparative tests using larger training and test corpora.*). In addition to these tests, a third experiment was performed that was designed to measure the relative performance of all three Baby implementations and other leading methods of processing the same data. Once again, performance indications were noted (see section *7.3 Results of comparative tests with other research models.*).

9.1.8 Advanced language behaviour.

Though the performance of Baby implementations showed promise, they were capable of processing a correct response to a novel utterance only if the resolution of processing the response relied upon the transformation of common patterns of language behaviour. This restriction limited the language domain of Baby models presented so far. In order for them to compete with other NLP implementations, it was necessary for this issue to be investigated.

It was hypothesised that whilst coincident patterns between previously experienced language behaviour may contain information that allowed Baby models to process language knowledge, information contained outside such patterns (variable elements) of the previously experienced language behaviour may allow Baby models to process appropriacy. Such an hypothesis would enable the contextualisation of utterances within the coherence of the current discourse. The original underpinning theory did not cater for this aspect, and was consequently considered insufficiently explanative. An augmented underpinning theory that encompassed these new concepts may be seen in section *8.2 FuzzyBaby.*

An externalisation of the augmented theory was constructed and the name FuzzyBaby was coined. This implementation modelled the notion that patterns of common behaviour related to the concept represented by the words "most closely correspond", and variable elements of patterns of behaviour related to the concepts represented by the words "contextually most similar". A third experimental hypothesis was developed to test this notion (see section *8.2.1 Rationale.*).

This hypothesis was tested in a less rigorous manner than that applied to the two main experimental hypotheses. This was because a proper investigation of the issues involved was considered too large to be encompassed in this research project. Consequently, their presentation in this thesis was intended for the purpose of illustrating the model's potential for future research. A model of the Baby concept that included fuzzy variable processing was designed, implemented and tested, and

performance indications were noted (see section *8.3 Some preliminary research that investigates advanced Baby behaviour.*).

During the course of this research, several interesting Baby behaviours were observed. Two were reported in this thesis. Firstly, the ability to process text, from which white space had been removed (see section *7.5.4.1 Processing textual representations of language containing no white-space.*) and secondly, a possible correspondence between the language acquisition performance of Baby models and early human language behaviour (see section *8.3.2.1 Language acquisition behaviour.*).

9.2 General discussion.

There follows a discussion concerned with issues that are better considered, now that this research project has been described in its entirety.

9.2.1 Grounding.

One of the most important issues connected with the Baby method of processing human language, is that it does not model a conventional human construct of the nature of language. In common with connectionist approaches, Baby's underpinning theory is grounded in aspects of what is known about the *behaviour* of neural systems. In essence, Baby's underpinning theory implies that to some degree or other, human textual language behaviour may be simulated by processing the largest occurrences of textual stimuli.

Research into the Baby concept is at an embryonic stage. Performance measures shown here may be considered as a beginning of an investigation into the approach. Nonetheless, it can be said that the tests applied to various Baby models have been contrived to contain large elements of common character patterns and it is this, rather than any other factor, that has determined performance. It can be counter-argued, however, that such patterning is an intrinsic property of human language. Such issues would form the basis for important future research into the Baby concept of NLP. At any rate, from a current perspective of linguistic theory, the underpinning theory sits on shaky ground. In defence of this situation, support for the theory may be strengthened by considering the possibility that in developing high level theories of the nature of language, humans attribute undue complexity to models of non-human language processing. There have been many descriptions of the nature of language, and they have been expressed from an abstract perspective. In the author's opinion, at a basic level, language generation may simply be considered as the transformation of neural activity into efferent stimuli. Whereas, the comprehension of language may be considered as the transformation of afferent stimuli into neural activity.

One could illustrate this description of language by drawing an analogy with code breaking. Assume that one had access to many examples of coded information but not the cipher key. Also, one had access to a cipher machine whose complexity precluded analysis. What options are there available to build another cipher machine to allow communication via the code? There are at least three methods that one might adopt. Firstly, one might investigate the code to establish its nature and thereby infer a theory that underpinned the key. This theory could then be used to model, and thus externalise, a second machine. Secondly, one might try and recreate the machine, widget by widget, in order to manufacture a duplicate in ignorance of its mechanism. Or thirdly, one might investigate the behaviour of the existing machine and construct another machine, using different widgets, that behaved sufficiently like its contemporary to enable it to cipher and decipher the code. In the order of their presentation, these methods represent symbolic, connectionist, and Baby model methods of processing the code of human language.

If one were to accept this low level theory of language, then the Baby approach is intellectually more appealing. The theory permits the strategy of modelling neural behaviour, equally with the language theory and neural reconstruction techniques.

9.2.2 Behaviourist implications.

It may be currently unfashionable, but the author supports the view that humans are predominantly behavioural creatures. That is not to say that for every particular stimulus, their exists a programmed response for a given individual. If this were the case, then how could creativity be explained, for example? It seems to the author, that the ingredient missing, is an acceptance of the massive complexity of the human neural system and further, that its processes may not be deterministic. Current perception of a Skinner type 'black-box' is a system that maps, in a deterministic manner, each stimulus with a response pattern that, in the past, has been most rewarding. From the author's perspective, it would be better to consider the black-box as a system where input/output mappings are non-deterministic. Each response would be determined via a mapping that only becomes fully specified by the context surrounding its stimulus. One is left, of course, with the task of suggesting a mechanism that could create these contextually appropriate mapping functions for each novel stimulus. The Baby concept is seen as providing a theory with which to model such a mechanism. Such a system would be observably behavioural, but not fall strictly within the Behaviourist camp, because of the lack of a deterministic cause and effect mapping. Indeed, as the Baby concept hypothesises the mechanism of the black-box, from this perspective it could be considered to exist within the Cognitive Science paradigm (how fuzzy boundaries become).

An important consequence of this neural behaviour modelling approach is that the Baby concept is not necessarily limited to processing language behaviour. It is hypothesised that the Baby concept is capable of simulating other human behaviours, or indeed the behaviours of other non-human systems, that are sufficiently complex for rule set modelling to be inappropriate. Researching this prospect more fully may be a worthwhile pursuit.

9.2.3 Notable Baby meta-behaviour.

There follows a set of Baby meta-behaviours that are considered worthy of special note.

9.2.3.1 Potential for wide domain language processing.

In section *2.4 An introduction to natural language processing.*, the discipline was described from the perspective of language and knowledge domain size. One of the reasons for highlighting this aspect of NLP, is that the author hypothesises wide domain language processing as one of the most important potentials of Baby-based NLP.

The underpinning theory does not model a conventional concept of the nature of language. Consequently, models of the theory do not contain artefacts such as conventional lexicons, re-write rule databases or semantic representations, for example. Potentially, the language domain that externalisations of the underpinning theory might process is a function of the language to which the they are exposed. This potential is, of course, shared with connectionist methods, however, for wide language domains, network design is highly problematic in the foreseeable future. In this research, rather than conceiving a theory that models natural language, Baby's underpinning theory allows the complexity of language itself to supply a solution, by providing behaviour pattern information generated directly from the complexity of the domain. If, after much more research than is described in this thesis, it is found that the underpinning theory is tractable, then it is hypothesised that wide domain capabilities would automatically follow.

9.2.3.2 Correlation between constant patterning and variable content responses.

Consider a situation where FuzzyBaby has conducted a Pattern Extraction process upon a given training corpus, where meta-Features have been derived. Table 42 shows the FeatureParts and VariableParts of a typical FeatureCouplet and the FeatureParts and VariableParts of a typical Meta(1)Couplet, that have been taken from the "Who is hot?" example shown in section *8.2.3 A worked example.*

FeatureCouplet:	
][Who is][?][	//June//~//Peter//~
][is a][.][	/June/girl//~/Peter/boy//~
Meta(1)Couplet:	
]0[is]1[.]2[	/Jane/hot//~/Peter/dozing//~/Potato/lovely//~/Running/exersise//~
]3[s]4[.]2[	/It i/a delight//~/He i/a sleeper//~/Colin eat/them//~/Paul doe/it//~

Table 42: Example of the FeaturePart and VariablePart of a Feature and a Meta(1)Feature.

There exists, as a consequence of Baby processes, a negative correlation between meta level number and Constant length in the FeaturePart of Features, and a positive correlation between meta level number and VariableCluster length in the VariablePart of Features. This phenomenon is manifest in the above example.

Now suppose that a novel utterance is input, which exhibits a low correspondence with utterances that comprise the training set, e.g. **"Who's hot"**. In this instance, a higher fuzzy evaluation exists between the input utterance and the FeaturePart of the Meta(1)Couplet, than with the more humanly intuitive FeaturePart of one of the FeatureCouplets, i.e.][Who is][?][. Consequently, the Meta(1)Couplet is computed by FuzzyBaby to form a competitive response. It can be seen that the competitive response generated contains a smaller Constant element and a larger VariableCluster element, than if it were formed from the FeatureCouplet. It is hypothesised that this is a manifestation of appropriacy, inasmuch as FuzzyBaby is responding to an utterance with ambiguous pattern classification, by compensating with information outside the domain of that learned to support language knowledge. It would be interesting to conduct an experiment designed to test a possible correspondence between Baby and human behaviour in this respect.

9.2.3.3 Noise.

It was noted in section *7.5.2 Comparative tests using larger training and test corpora.*, that models of Baby showed improved performance when processing sentential representation of language, rather than single word representations of the same behaviour. It was explained that this phenomenon was due to a higher signal to noise ratio (Constant to Variable ratio) present in sentential representations. Consequently, it was hypothesised that this phenomenon might be a model-intrinsic behaviour. Such behaviour can also be seen as an example of a possible correspondence between Baby and human language behaviour. No research can be found that tests this hypothesis, but it would be an interesting research project to pursue, always assuming that an

ethical experiment could be devised. It seems difficult to imagine that children could learn past tense behaviour so well, merely by exposure to verb only experience.

Another interesting research project, that would be easier to design and perform, would be to investigate the performance of ANNs when processing sentential representations of language behaviour, as opposed to word only association. Again, in the case of past tense learning, no published research could be found.

9.2.3.4 Non-hypothesisable utterances.

Table 35: Set of data that maps the process of fuzzy Constant and fuzzy Variable information generated in the worked example., illustrates that Baby models hypothesise incomplete responses along with full responses. The string "(...)" denote such occurrences. Non-hypothesisable utterances are generated when all the variable symbols in the response SignaturePart of a FeatureCouplet do not appear in the stimulus SignaturePart of FeatureCouplets. The reason for generating non-hypothesisable responses, was to render the processes involved with early Baby models transparent. However, it should be noted that it also renders them less brittle by allowing best guess incomplete solutions to be hypothesised. Testing the correlation between Baby and human language behaviour in this respect would also be an interesting research project.

9.2.3.5 Scaling

The issue of computational tractability in scaling NLP systems to cope with larger language domains is generally problematic (this is seen as a different problem than the language theoretical issues discussed in *9.2.3.1 Potential for wide domain language processing.*). For the purpose of the research described in this thesis, computational tractability of Baby models was not addressed. This is because of the embryonic stage of the work. Baby models were implemented in such a way as to investigate the basic behaviour of externalisations of the underpinning theory. It should be noted however, that there is a compression of data associated with the Pattern Extraction process. This is because FeatureCouplets characterise utterance structure in much the same way as, for example, parts of speech characterise word function in Chomskyean type grammars. Also, it should be noted that the Pattern Extraction process is more computationally expensive than Conversationally Interactive processing. For example, in the case of the 1000 verb sentential training corpus, Pattern Extraction is between three and four orders of magnitude slower than Conversationally Interactive processing when producing individual responses from the patterns extracted.

9.2.3.6 Small training set learning.

In most other methods of machine language acquisition, significant learning is observed only after exposure to large amounts of training. It can be seen from results shown in section *7.2 Results of comparative tests using larger training and test corpora.*, that Baby models require relatively less exposure for significant acquisition to take place. If this phenomenon proves to be model intrinsic, then the behaviour may be beneficial. Bearing in mind the possibility that Baby models may operate in domains other than language, (see *9.2.2 Behaviourist implications.*), they would have the capability of acquiring aspects of the behaviour of limited sized data sets. This phenomenon is often observed with experimental results for example.

9.2.3.7 Correlation between the Pattern Extraction process and human sleeping.

Finally, it should be noted that there may be similarities between the Pattern Extraction process and some aspects of theories that seek to explain the phenomenon of human sleep. During Pattern Extraction, models of Baby detect patterns between recently experienced TextCouplets. If such patterns have already been detected then their Variable elements are added to corresponding FeatureCouplets. If such patterns are novel then new FeatureCouplets are added to the FeatureCouplet/MetaCouplet databases. In either case, Pattern Extraction enhances Baby's language processing abilities by producing FeatureCouplets that may be associated with newly experienced stimuli.

Similar types of mechanism have also been observed to occur during sleep. For example, Hennevin, Hars, Maho & Bloch (1995) described three sets of experiments into cortical activity in rats during random eye movement sleep and found that amongst other things they:

- form new associations and respond to them in subsequent wake;
- reinforce prior learning to enhance subsequent waking behaviour.

More generally, considerable evidence now supports the critical role of sleep in the processing of memory. However, rather than simply strengthening established memories, sleep appears to serve two distinct functions:

- to support the transfer of memories from the hippocampus to the neocortex;
- to foster the integration of neocortical memories into wider associative networks.

(Stickgold, 1998)

At a simplistic level, one of the functions of the hippocampus is to process newly learned information (Kandel & Hawkins, 1993), whilst one of the functions of the neocortex is to "draw on visual and other experiences, to rewire itself to create categories or features it can respond to" (Crick and Koch, 1993). Loosely, one might correlate the functions of the TextCouplet database with the hippocampus, the Feature and MetaFeature databases with the neocortex, and the Pattern Extraction process with sleep.

These correlations are contentious issues and may form the basis for an interesting research project.

9.3 Conclusions.

In this work, various hypothesise have been posited. They were designed to represent significant sub-domains of Baby's underpinning theory. This section concludes upon the extent to which the results of test experiments have supported or not supported them.

9.3.1 General tractability of the Baby method of NLP (the first hypothesis).

An experiment was designed to test the first hypothesis. The experiment subjected NonfuzzBabe to five different types of simple language interaction: subject-verb agreement; past tense formation; plural formation; bi-lingual transformation and passive transformation. Two further experiments were conducted to test the model's ability to distinguish between both different types of plural formation, and between passive transformation and past tense transformation that are present in the one training set. From the results obtained from all seven tests, it was concluded that hypothesis one had been supported (see section *5.0 Results of NonfuzzBabe's language performance.*).

9.3.2 The tractability of associating fuzzy processes to models of Baby (the second hypothesis).

An experiment was performed to test the second hypothesis by employing the seven simple language interaction types above to CorrBabe and FuzzBabe. It was concluded that within the domain of these tests, Baby models implemented with fuzzy classification techniques significantly out performed Baby models implemented with equivalence classification techniques (see section *7.1 Results of comparative tests with NonfuzzBabe.*). These results were interpreted as supporting the second experimental hypothesis.

Irrespective of this interpretation, it was thought that these tests provided inconclusive support for the second hypothesis, because of their limited size. Consequently, more extensive experiments were designed to test one of the simple interaction types more rigorously. Past tense formation was chosen because (amongst other issues) much research had be conducted in this area by other

researchers. The experiments were designed to test three aspects of NonfuzzBabe, CorrBabe and FuzzBabe performance. The aspects were training accuracy, the prediction of seen verbs and the prediction of unseen verbs. Each aspect was tested in two conditions. Firstly, using sentential representations of the formation of past tense and secondly, using verb only representation of the formation of past tense.

The training and test corpora employed with these tests were of significant size and the results allowed a conclusion to be drawn that, for the language interaction used in the tests, Baby models implemented using fuzzy classification techniques, produced superior language processing performance compared with a Baby model implemented using equivalence classification techniques (see section *7.2 Results of comparative tests using larger training and test corpora.*).

10.3.3 Processing of advanced language behaviour (the third hypothesis).

An experiment was performed to test the third hypothesis. Processing advanced language behaviour required implementations to process language appropriacy as well as language knowledge.

Firstly, a test was designed that comprised textual examples contrived to contain ambiguity, incomplete textual information and pronominal reference. It was shown that within the domain of this contrived test, FuzzyBaby was able to process linguistic ambiguity whilst NonfuzzBabe, CorrBabe and FuzzBabe were not (see section *8.2.3 A worked example.*).

Secondly, results of a pilot study designed to apply FuzzyBaby to a commercial reading comprehension test were shown. A measure of comprehension was observed, whereas it could not with NonfuzzBabe, CorrBabe and FuzzBabe (see section *8.3 Some preliminary research that investigates advanced Baby behaviour.*).

It may be concluded from these results that they provided sufficient support for the third hypothesis, for it to be tested more rigorously.

9.4 Finale.

In this thesis, research has been described that pursued an alternative approach to the modelling of AI machines. The domain chosen to test the alternative approach was NLP and the generic term Baby was coined for the approach. The research comprised an investigation into processing character patterns occurring in textual representations language.

As part of the introduction, a series of goals was identified that could be used to judge the success of this research, they were:

- to indicate whether a low-fuzzy model of the underpinning theory is capable of processing simple language interaction, as judged by a human user of the language;
- to indicate the extent to which the incorporation of fuzzy processes to the model, effect its simple language interaction performance, as judged by a human user of the language;
- to measure the relative performance between models of the underpinning theory and other methods of processing the same language behaviour;
- to indicate the performance of a model of the underpinning theory when applied to more complex language interaction.

It is considered that based upon these measures, the research has achieved its objectives.

10.0 References.

Allen, J. (1995). Natural Language Understanding, The Benjamin/Cummings Publishing Company, Inc.

Allen, R. B. and M. E. Riecken (1988). Anaphor and Reference in Connectionist Language Users. In Proceedings of: *The International Computer Science Conference,* Hong Kong .

Alshawi, H., Ed. (1992). The Core Language Engine. Cambridge, MA, The MIT Press.

Alvarado, S. J. (1992). Argument Comprehension. In (ed. S. C. Shapiro): Encyclopaedia of Artificial Intelligence second edition., New York, John Wiley.

Anderson, E. S., Dunlea, A. & Kekelis, L. (1993). "The Impact of Input: language acquisition in the visually impaired Special issue: language development in special populations." First-Language., Vol. 13, (37): , pp. 23 - 49.

Anderson, J. R. (1977). "Induction of Augmented Transition Networks." Cognitive Science., Vol. 1, : , pp. 125 - 157.

Anderson, J. R. (1990). Cognitive Science and its Implications, WH Freeman & Co.

Armstrong, S., Russell, G., Petitpierre, D. & Robert, G. (1995). An Open Architecture for Multilingual Text Processing. In Proceedings of: *SIGDAT95 (EACL-95 workshop)*, 30 - 34 Dublin .

Bar-Hillel, Y., Ed. (1960). The Present Status of Automatic Translation of Languages. Advances in Computers. New York, Academic Press.

Barnett, J. A. (1981). Computational methods for a mathematical theory of evidence. In Proceedings of: *IJCAI'87*, 868 - 875 Vancouver, BC.

Beale, R. and T. Jackson (1992). Neural Computing: an introduction., Bristol, Institute of Physics Publishing.

Biederman, I. (1987). "Recognition by components: a theory of human understanding." Psychological Review., Vol. 94, : , pp. 115 - 147.

Bod, R. (1992). Data Oriented Parsing (DOP). In Proceedings of: *COLING 92,* Nantes .

Bod, R. (1995). Enriching Linguistics with Statistics: performance models of natural language. In (ed. ILLC Dissertation Series 1995-14., University of Amsterdam.

Bod, R. (1996a). Efficient Algorithms for Parsing the DOP model? A reply to Joshua Goodman. In Proceedings of: *cmp-lg/9605031,* .

Bod, R. (1996b). Two Questions about Data-Oriented-Parsing. In Proceedings of: *Fourth Workshop on Very Large Corpora cmp-lg/9606022,* Copenhagen, Denmark .

Bod, R. and R. Scha (1996c). "Data-Oriented Language Processing: An Overview," ILLC Research Report LP-96-13, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands.

Boden, M. (1977). Artificial Intelligence and Natural Man, The Harvester Press Ltd.

Bonissone, P. P. (1992). Reasoning, Plausible. In (ed. S. C. Shapiro): Encyclopaedia of Artificial Intelligence Second Edition., New York, John-Wiley.

Bonissone, P. P., Gans, S. S. & Decker, K. S. (1987). RUM: A layered Architecture for Reasoning with Uncertainty. In Proceedings of: *IJCAI'87,* 373 - 379 Milan, Italy .

Bonnema, R.(1996) *Data-Oriented Semantics* Ph.D. Thesis, University of Amsterdam

Brown, R. (1973). A First Language., Cambridge, MA, Harvard University Press.

Charniak, E. (1996a). "Tree-bank Grammars," Technical Report CS-96-02, Department of Computer Science, Brown University, Rode Island.

Charniak, E. (1996b). Tree-bank Grammars. In Proceedings of: *AAAI'96,* Portland, Oregon .

Cheeseman, P. (1985). In Defense of Probability. In Proceedings of: *IJCAI,* 1002 - 1009 Los Angeles .

Chomsky, N. (1965). Aspects of the Theory of Syntax, MIT Press.

Clark, D. A.(1988a) *Psychological Aspects of Uncertainty and their Implications for Artificial Intelligence* Ph.D. Thesis, University of Wales

Clark, D. A. (1990). "Verbal Uncertainty Expressions." Current Psychology: research and reviews., Vol. 9, (3): , pp. 203 - 235.

Clark, M. R. B. (1978). Negation as Failure. In (ed. H. Gallaire and J. Minker): Logic and Databases., New York., Plenum Press.

Collier, R. (1994). "An Historical Overview of Natural Language Processing Systems that Learn." Artificial Intelligence Review., Vol. 8, (1): , pp. 17 - 54.

Cottrell, G. W. (1989). A Connectionist Approach to Word Sense Disambiguation, Pitman Publishing.

Coulthard, M., Ed. (1992). Advances in Spoken Discourse Analysis, Routledge.

Crain, S. (1991). "Language acquisition in the absence of experience." Behavioural and Brain Sciences., Vol. 14, (4): , pp. 597-612.

Crick, F. and C. Koch (1993). The Problem of Consciousness. In (ed. J. Piel): Mind and Brain., W. H. Freeman & Company: 125 - 136.

Cutting, D., Kupiec, J., Pedersen, J. & Sibun, P. (1992). A Practical Part-of-Speech Tagger. In Proceedings of: *Third Conference on Applied Natural Language Processing, (ANLP-92)*, 133 - 140 Trento .

Dempster, A. P. (1968). "A Generalisation of Bayesian inference." Journal of the royal statistical society, Series B., Vol. 30, : , pp. 205 - 247.

Dennett, D. C. (1984). Cognitive Wheels: the frame problem in AI. In (ed. C. Hookway): Minds Machines and Evolution: philosophical studies., Cambridge, Cambridge University Press.

Diaper, D. (1986). Identifying the Knowledge Requirements of an Expert System's Natural Language Processing Interface. In (ed. M. D. Harrison and A. F. Monk): People and Computers: Designing for Usability., Cambridge University Press.

Diaper, D. and T. Shelton (1987). Natural Language Requirements for Expert System Naive User. In (ed. Recent Developments and Applications of Natural Language Understanding., Unicom Seminars Ltd.

Dimcovic, N. and M. J. Tobin (1995). "The use of Language in Simple Classification Tasks by Children who are Blind." Journal of Visual Impairment & Blindness., Vol. 89, (5): , pp. 448 - 459.

Dreyfus, H. L. and D. S.E. (1986). Mind Over Machine, Free Press.

Dubois, D. and H. Prade (1982). On several representations of an uncertain body of evidence. In (ed. M. M. Gupta and E. Sanchez): Fuzzy Information and Decision Processes., Amsterdam, North Holland: 167 - 181.

Duda, R., Gashnig, J. & Hart, P. (1979). Model design in the Prospector consultant system for mineral exploitation. In (ed. D. Nichie): Expert Systems in the Microelectronic Age., Edinburgh, Edinburgh University Press.

Edleman, G. M., Ed. (1985). Neural Darwinism. How We Know, Harper and Row.

Elman, J. L. (1990). "Finding structure in time." Cognitive Science., Vol. 14, (2): , pp. 179 - 211.

Ervin, S. (1964). Imitation and Structural Change in Children's Language. In (ed. E. Lenneberg): New Directions in the Study of Language., Cambridge MA, MIT Press.

Feldman, J., Lakoff, G., Bailey, D., Narayanan, S., Regier, T. & Stockle, A. (1996). "Lo The First Five Years of an Automated Language Acquisition Project." Artificial Intelligence Review., Vol. 10, : , pp. 103 - 129.

Fischbach, G. D., Ed. (1993). Mind and Brain. Readings from Scientific American, Freeman and Company.

Fox, J. (1986). Knowledge, Decision Making and Uncertainty. In (ed. W. A. Gale): Artificial Intelligence and Statistics., Reading, Massachusetts, Addison-Wesley.

Gasser, M. E. (1988). "A Connectionist Model of Sequence Generation in a First and Second Language," Technical Report UCLA-A1-88-13, Artificial Intelligence Laboratory, Computer Science Department, University of California.

Giachin, E., Lee, C. H., Rabiner, L.R., Rosenberg, A.E. & Pieraccini, R. (1992). "On the use of inter-word context-dependent units for word juncture modelling." Computer Speech and Language., Vol. 6, : , pp. .

Gold, E. (1967). "Language Identification in the Limit." Information and Control., Vol. 16, : , pp. 447 - 474.

Goodman, J. (1996). Efficient Algorithms for Parsing the DOP model. In Proceedings of: *Empirical Methods in Natural Language Processing,* Philadelphia, PA .

Gorin, A. (1995). "On Automated Language Acquisition." Journal of the Acoustical Society of America., Vol. 97, (6): , pp. 3441 - 3461.

Granger, R. H. and P. Foulu (1977). FOULUP. In Proceedings of: *5th International Joint Conference on Artificial Intelligence,* 172 - 178.

Greenberg, J. H. (1963). Universais of Language., Cambridge MA, MIT press.

Grosz, B. J., Appelt, D., Martin, P. & Pereira, F. (1987). "TEAM: an experiment in the design of transportable natural-language interfaces." Artificial Intelligence., Vol. 32, (2): , pp. 173 - 244.

Hanson, S. J. and J. Kegl (1987). PARSNIP: A Connectionist Network that Learns Natural Language Grammar from Exposure. In Proceedings of: *The Ninth Annual Conference of the Cognitive Science Society,* 106 - 119 Seattle, Washington Lawrence Erlbaum.

Hedrick, C. (1976). "Learning Production Systems from Examples." Artificial Intelligence., Vol. 7 : pp. 21 - 49.

Hellendoorn, H. and C. Thomas (1993). "Defuzzification in Fuzzy Controllers." Intelligent and Fuzzy Systems., Vol. 1, : , pp. 109 - 123.

Hendrix, G., Sacerdoti, D., Sagalowiez, & Slocum, J. (1978). "Developing a Natural Language Interface to Complex Data." ACM Trans on Database Systems., Vol. 3, (2): , pp. 105 - 147.

Hennevin, E., Hars, B., Maho, C. & Bloch, V. (1995). "Processing of Learned Information in Paradoxical Sleep: relevance for memory." Behavioural Brain Research., Vol. 69, (1 - 2): , pp. 125 -135.

Hill, J. C. (1983). "A Computational Model of Language Acquisition of the Two Year Old." Cognition and Brain Theory., Vol. 6, (3): , pp. 287 - 317.

Hobbs, J. R., Croft, W., Davies, T., Edwards, D. & Laws, K (1987). "Commonsense Metaphysics and Lexical Semantics." Computational Linguistics., Vol. 13, (3 - 4): , pp. 241 - 250.

Holland, J.H., Holyoak, K.J., Nisbett, R.E. and Thagard, P.R. (1986). Induction, The MIT Press.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. In Proceedings of: *National Academy of Science*, 2554 - 2558 USA .

Jacob, P. and L. Rau (1990). "SCISOR: a system for extracting information from on-line news." Communications of the ACM., Vol. 33, (11): , pp. 88 - 97.

Jang, J. R., Sun, C. & Mizutani, E. (1997). Neuro-Fuzzy and Soft Computing: a computational approach to learning and machine intelligence, Prentice Hall.

Jelinek, F., Lafferty, J. D. & Mercer, R.L. (1990). "Basic Methods of Probabilistic Grammars," Technical report IBM, Yorktown Heights, NY.

Joshi, A. K. (1991). "Natural Language Processing." Science., Vol. 11, (4): , pp. 1242 - 1249.

Joshi, A. K. (1993). "Natural Language Processing, some recent trends." Current Science., Vol. 64, (6): , pp. 393 - 406.

Kahneman, D., Slovic, P. & Tversky, A. Eds. (1982). Judgment under uncertainty: heuristics and biases. New York, Cambridge University Press.

Kandel, E. R. and R. D. Hawkins (1993). The Biological Basis of Learning and Individuality. In (ed. J. Piel): Mind and Brain., W. H. Freeman & Company: 40 - 53.

Kaplan, R. (1996). A Probabilistic approach to Lexical-Functional Grammar. In Proceedings of: *slides of the keynote lecture held at LFG workshop 1996*, Grenoble .

Kelley, K. L. (1967). "Early Syntactic Acquisition," Technical Report No. P-3179, Rand Corporation, California.

Kolodner, J. L. (1980). "Retrieval and Organisational Strategies in Conceptual Memory: a computer model," Technical report 187, Yale University, New Haven, Connecticut.

Krause, P. and D. Clark (1993). Representing Uncertain Knowledge: an artificial intelligence approach, Kluwer Academic Publishers.

Kuczaj, S. A. (1977). "The Acquisition of Regular and Irregular Past Tense Forms." Journal of Verbal Learning and Verbal Behaviour., Vol. 16, : , pp. 589 - 600.

Lakoff, G. (1986). Women, Fire and Dangerous Things: what categories reveal about the mind, University of Chicago Press.

Langley, P. (1982). "Language Acquisition Through Error Recovery." Cognition and Brain Theory., Vol. 5, (3): , pp. 211 - 255.

Langley, P. and R. T. Neches (1981). "PRISM User's Manual," Technical Report Department of Computer Science, Carnegie-Mellon University, Pennsylvania.

Lefley, M. (1993). How SEARS works, Personal Communication.

Lefley, M. (1995). The SEARS project, Personal Communication.

Lefley, M. and P. Rogers (1997). Using Patterns in State Spaces of Complex Problems to Hypothesise Solution Behavior. In Proceedings of: *Eighth Ireland Conference on Artificial Intelligence (AI-97),* Dublin.

Lefley, M. and P. Rogers (1998). Our Fuzzy Baby: using fuzzy competitive logic to learn natural language interpretation. In Proceedings of: *International Symposium on Engineering of Intelligent Systems EIS98,* Tenerife, Spain .

Lefley, M. and P. Rogers (1998a). Baby: a system that learns natural language processing by using pattern recognition and fuzzy competitive mechanisms. In Proceedings of: *Recent Advances in Soft Computing,* Leicester, UK.

Lefley, M. and P. Rogers (1999). Incorporating Fuzzy Resonance in Language Acquisition and Production. In Proceedings of: *CIMA,* New York .

Lehman, W. P. (1962). Historical Linguistics: an introduction., New York, Holt, Rinehart and Winston.

Lifschitz, V. (1985). "Closed world databases and circumscription." Artificial Intelligence., Vol. 27, (2): , pp. 229 - 235.

Lindsay, R. K., Ed. (1963). Inferential Memory as the Basis for Machines which Understand Natural Language. Computers and Thought. New York, McGraw-Hill.

Ling, C. X. (1994). "Learning the Past tense of English Verbs: the symbolic pattern associator Vs Connectionist models." Journal of Artificial Intelligent Research., Vol. 1, : , pp. 209 - 229.

Liu, R. L. and V. W. Soo (1994). "Explanation-Based Natural-Language Acquisition using Universal Linguistic Principles as Innate Domain Theory." Applied Artificial Intelligence., Vol. 8, (4): , pp. 459 - 481.

Locke, W. N. and A. D. Booth (1955). Machine Translation of Languages: fourteen essays., Cambridge, Massachusetts, The MIT Press.

MacWhinney, B. (1999). MacWhinney verb corpus.,

MacWhinney, B. and J. Leinbach (1991). "Implementations are not conceptualizations: Revising the verb model." Cognition., Vol. 40, : , pp. 73 - 193.

MacWhinney, B. (1987). The Competition Model. In (ed. B. MacWhinney): Mechanisms of Language Acquisition., Hillsdale, NJ., Lawrence Erlbaum.

Manaris, B. Z. and B. S. Slator (1996). "Interactive Natural Language Processing: building on success." Computer., Vol. 29, (7): , pp. 28-31.

Margolis, H. U. o. C. P. (1987). Patterns, Thinking and Cognition: a theory of judgment, University of Chicago Press.

Markowitz, J. A. (1996). Using Speech Recognition, Prentice Hall.

Marr, D. (1982). Vision, WH Freeman & Co.

McCarthy, J. (1980). "Circumscription: a form of non-monotonic reasoning." Artificial Intelligence., Vol. 13, (1 - 2): , pp. .

McCarthy, J. (1986). "Applications of circumscription to formalising common sense knowledge." Artificial Intelligence., Vol. 28, (1): , pp. 89 -116.

McConachie, H. (1990). "Early Language Development and Severe Visual Impairment." Child Care, Health and Development., Vol. 16, (1): , pp. 55 - 61.

McConachie, H. R. and V. Moore (1994). "Early Expressive Language of Severely Visually-Impaired Children." Developmental Medicine and Child Neurology., Vol. 36, (3): , pp. 230 - 240.

McCulloch, W. S. and W. Pitts (1943). "Bull Math." Biophysics., Vol. 5, : , pp. 115 - 133.

McDermott, D. and J. Doyle (1980). "Non-monotonic logic." Artificial Intelligence., Vol. 13, (1 - 2): , pp. .

McKevitt, P. (1992). "Approaches to Natural Language Discourse Processing." Artificial Intelligence Review., Vol. 6, (4): , pp. 327 - 311.

McKevitt, P., Ed. (1996). Integration of Natural Language and Vision Processing (volume iv): recent advances, Kluwer.

Meier, P. R. (1991). "Language Acquisition by Deaf Children." American Scientist., Vol. 79, (1): , pp. 60 - 70.

Michalski, R. (1992). Understanding the Nature of Learning: issues in research directions. In (ed. R. S. Michalski, J. G. Carbonell and T. M. Mitchell): Machine Learning: an artificial intelligence approach., 2. California, Morgan Kaufmann: 3 - 25.

Minsky, M. (1975). A Framework for Representing Knowledge. In (ed. P. H. Winston): The Psychology of Computer Vision., New York, McGraw-Hill: 211 - 217.

Mooney, R. (1985). "Generalising Explanations of Narratives into Schemata," Technical Report No. T-147, University of Illinois (Coordinated Science Laboratory), Illinois.

Nauck, D., Klawonn, F. & Kruse, R. (1997). Foundations of Neuro-Fuzzy Systems, John Wiley & Sons.

Newell, A. and H. A. Simon (1976). "Computer Science as an Empirical Inquiry: symbols and search." Communications of the ACM., Vol. 19, (3): , pp. 113-126.

Ney, H., Essen, U. & Kneser, R. (1994a). "On Structuring Probabilistic Dependencies in Stochastic Language Modelling." Computer Speech and Language., Vol. 8, (1): , pp. 1 - 38.

Ney, H., Steinbiss, V., Haeb-Umbach, R., Tran, B.H. & Essen, U. (1994). "An Overview of the Philips Research System for Large-Vocabulary Continuous-Speech Recognition." International Journal of Pattern Recognition and Artificial Intelligence, Special Issue on Speech Recognition for Different Languages., Vol. 8, (1): , pp. 33 - 70.

Obermeier, K. K. (1989). Natural Language Technologies in Artificial Intelligence, Ellis Horwood.

Orponen, P. (1990). "Dempster's Rule is #P-complete." Artificial Intelligence., Vol. 44, : , pp. 245 - 253.

Parsons, S. (1992). Qualitative Belief Networks. In Proceedings of: *ECIA '92*, 48 - 50 Vienna, Austria .

Parsons, S. and E. H. Mamdani (1993). On Reasoning in Networks with Qualitative Uncertainty. In Proceedings of: *9th Conference on Uncertainty in Artificial Intelligence*, Washington DC .

Passino, K. M. and S. Yurkovich (1996). Fuzzy Control., California, Addison Wesley Longman, Inc.

Pereira, F. (1983). "Logic for Natural Language Analysis," Technical Note. 275, SRI International.

Perezpereira, M. (1994). "Imitations, Repetitions, Routines, and the Child's Analysis of Language: insights from the blind." Journal of Child Language., Vol. 21, (2): , pp. 317 - 337.

Pollack, J. B. (1990). "Recursive Distributed Representations." Artificial Intelligence., Vol. 46, : , pp. 77-105.

Prasada, S. and S. Pinker (1993). "Generalization of regular and irregular morphological patterns." Language and Cognitive Processes., Vol. 8, (1): , pp. 1 - 56.

Quinlan, E. and S. O'Brian (1992). Sublanguage: characteristics and selection guidelines for MT. In Proceedings of: *AI and Cognitive Science '92: annual Irish conference on artificial intelligence and cognitive science*, 342 - 345 Limerick, Ireland Springer-Verlag.

Quinlan, J. (1993). C4.5 Programs for machine learning., San Mateo, CA., Morgan Kaufmann.

Rajman, M.(1995a) Apports d'une approche a base de corpus aux techniques de traitement automatique du language naturel Ph.D. Thesis.

Rajman, M. (1995b). "Approche Probabiliste de l'Analyse Syntaxique." Traitement Automatique des Langues., Vol. 36, (1-2): , pp. .

Reeker, L. H. (1976). "The Computational Study of Language Acquisition." Advances in Computers., Vol. 15, : , pp. 181 - 237.

Reilly, R. G. and N. Sharkey, Eds. (1992). Connectionist Approaches to Natural Language Processing, Earlsdale.

Reiter, R. (1978). On closed world data bases. In (ed. H. Gallaire and J. Minker): Logic and data bases., New York, Plenum Press.

Reiter, R. (1980). "A logic for default reasoning." Artificial Intelligence., Vol. 13, (1 - 2): , pp. .

Reiter, R. (1984). Towards a Logical Reconstruction of Relational Database Theory. In (ed. M. Brodie, J. Myopoulos and J. W. Schmidt): On Conceptual Modelling., New York, Springer-Verlag: 163 - 189.

Rich, E. and K. Knight (1991). Artificial Intelligence, McGraw-Hill, Inc.

Rogers, P. and M. Lefley (1997a). "Processing Patterns in Text: an alternative approach to natural language processing." The European Students Journal of Language and Speech., Vol. 1, (#97/01)

Rogers, P. and M. Lefley (1999). The Baby Project: an alternative approach to machine learning of wide domain natural language environments. In (ed. Y. A. Wilks): Machine Conversations., Boston, Kluwer Academic Press.

Rogers, P. and M. Lefley (1997). The Baby Project: an alternative approach to machine learning of wide domain natural language environments. In Proceedings of: *First International Workshop on Human-Computer Conversation,* Bellagio .

RosenBlatt, F. (1962). Principles of Neurodynamics: perceptrons and the theory of brain mechanisms., Washington, DC., Spartan Books.

Ross, T., J. (1995). Fuzzy Logic with Engineering Applications, McGraw-Hill, Inc.

Rumelhart, D. E. and J. L. McClelland (1986). On Learning the Past Tense of English Verbs. In (ed. J. E. McClelland and D. E. Rumelhart): Parallel Distributed Processing: psychological and biological models., 2. Cambridge, MA., MIT Press.

Rumelhart, D. E., Hinton G. E. and Williams, R. J. (1986a). Learning Internal Representations by Error Propagation. In (ed. D. E. Rumelhart and J. L. McClelland): Parallel Distributed Processing: explorations in microstructures of cognition., 1. Cambridge, MA., MIT Press.

Rumelhart, D. E., Hinton G. E. and Williams, R. J. (1986b). "Learning Internal Representations by Back-Propagating Errors." Nature., Vol. 323, : , pp. 533 - 536.

Russell, S. and P. Norvig (1995). Artificial Intelligence: a modern approach, Prentice Hall.

Salveter, S. (1979). "Inferring Conceptual Graphs." Cognitive Science., Vol. 3, (2): , pp. 141 - 166.

Scha, R. (1990). Taaltheorie en Taalthechnoligie; competence and performance. In (ed. Q. A. M. de Kort and G. L. J. Leerdam): Computertoepassingen in de Neerlandistiek., Landelijke Vereniging van Neerlandici.

Scha, R. (1992). "Virtuele Grammatica's en Creatieve Algoritmen." Gramma/TTT., Vol. 1, (1):

Schank, R. C. (1973). Identification of Conceptualizations Underlying Natural Language. In (ed. R. C. Schank and K. M. Colby): Computer Models of Thought and Language., San Francisco, Freeman.

Schank, R. C. and R. Abelson (1977). Scripts, Plans, Goals and Understanding., New Jersey, Lawrence Erlbaum.

Schubert, L. K. and F. J. Pelletier (1982). "From English to Logic: context free computation of conventional logical translation." AJCL., Vol. 8, (1): , pp. 165 - 176.

Sejnowski, T. J. and C. R. Rosenburg (1987). "Parallel Networks that Learn to Pronounce English Text." Complex Systems., Vol. 1, pp. 145 - 168.

Sekine, S. and R. Grishman (1995). A Corpus-Based Probabilistic Grammar with Only Two Non-Terminals. In Proceedings of: *Fourth International Workshop on Parsing Technologies*, Prague, Czech Republic .

Selfridge, M. (1980). "A Process Model of Language Acquisition," Computer Science Technical Report 172, Yale University, New Haven, Connecticut.

Selfridge, M. (1981). A Computer Model of Child Language Acquisition. In Proceedings of: *7th International Joint Conference on Artificial Intelligence*, 92 - 96 .

Selfridge, O. G. (1959). Pandemonium: a paradigm for learning. In (ed. D. V. Blake and A. M. Uttley): Proceedings for the symposium on Mechanization of Thought Processes., National Physical Laboratory, Teddington, U.K., Her Majesty's Stationary Office: 511 - 529.

Selfridge, O. G. and U. Neisser (1960). "Pattern Recognition by Machine." Scientific American., Vol. 203, : , pp. 60 - 68.

Shafer, G. (1976). A Mathematical Theory of Evidence, Princeton University Press.

Shafer, G., Shenoy, P. P. & Mellouli, K. (1987). "Propagating Belief Functions in Qualitative Markov Trees." Int J Approximate Reasoning., Vol. 1, : , pp. 349 - 400.

Sharkey, N. E., Ed. (1988). A PDP System for Goal-Plan Decision. Cybernetics and Systems. Dordrecht, The Netherlands, Kluwer Academic.

Sharples, M., Hogg, D., Hutchinson, C., Torrance, S. & Young, D. (1990). Computers and Thought: a practical introduction to artificial intelligence, The MIT Press.

Shenoy, P. P. and G. Shafer (1986). Propagating Belief Functions with Local Computations. In Proceedings of: *IEEE expert*, 43 - 52 .

Shenoy, P. P. and G. Shafer (1990). Axioms for Probability and Belief Function Propagation. In (ed. R. Shachter, T. Levitt, J. Kanal and J. Lemmer): Uncertainty in Artificial Intelligence 4., North Holland, Elservier Science Publishers.

Siklossy, L. (1972). Natural Language Learning by Computer. In (ed. H. A. Simon and L. Siklossy): Representation and Meaning: experiments with information processing systems., Eaglewood Cliffs, New Jersey, Prentice Hall.

Sima'an, K. (1996). Computational Complexity and Probabilistic Disambiguation by means of Tree Grammars. In Proceedings of: *COLING - 96,* Copenhagen .

Sima'an, K., Bod, R., Krauwer, S. & Scha, R. (1994). Efficient Disambiguation by means of Stochastic Tree Substitution Grammars. In Proceedings of: *International Conference on New Methods in Language Processing,* UMIST, Manchester .

Sima'an, K. (1995). An optimized algorithm for Data Oriented Parsing. In Proceedings of: *International Conference on Recent Advances in Natural Language Processing,* Tzigov, Bulgaria .

Siskind, J. M. (1990). Acquiring Core Meaning of Words, Represented as Jackendoff-Style Conceptual Structures, from Correlated Streams of Linguistic and Non-Linguistic Input. In Proceedings of: *28th Annual Meeting of the Association for Computational Linguistics,* 143 - 156 .

Skinner, B. F. (1957). Verbal Behaviour., New York, Appleton - Century - Crofts.

Smyth, M. M., Collins, A. F., Morris, P.E. & Levy, P. (1994). Cognition in action, LEA.

Sommerville, I. (1989). Software Engineering 3rd Edition, Addison-Wesley Publishing Company.

Sparck-Jones, K. and R. J. Galliers (1995). Lecture Notes in Artificial Intelligence 1083., Berlin, Heidelberg, Springer-Verlag.

Stickgold, R. (1998). "Sleep: off-line memory reprocessing." Trends in Cognitive Science., Vol. 2, (12): , pp. 484 - 492.

Terano, T., Asai, K. & Sugeno, M. Eds. (1994). Applied Fuzzy Systems, AP Professional.

Touretzky, D. S. (1984). Implicit Ordering of Defaults in Inheritance Systems. In Proceedings of: *AAAI'84*, 322 - 325 Austin, Texas .

Tugwell, D. (1995). A State-Transition Grammar for Data-Orientated Parsing. In Proceedings of: *EACL '95*, Dublin, Ireland .

Turing, A. M. (1950). "Computing Machinery and Intelligence." Mind LIX., Vol. , (2236): , pp. 433 - 460.

Tversky, A. and D. Kahneman (1974). "Judgment under uncertainty: heuristics and biases." Science., Vol. 185, : , pp. 1124 - 1131.

van den Berg, M., Bod, R. & Scha, R. (1994). A Corpus Based Approach to Semantic Interpretation. In Proceedings of: *Ninth Amsterdam Colloquium*, Amsterdam, The Netherlands .

Vasconcellos, M. and M. Leon (1985). "SPANAM and ENGSPAN: machine translation at the Pan American Health Organisation." Computational Linguistics., Vol. 11, (2 - 3): , pp. 122 - 136.

Voorhees, E. M. (1993). Using Wordnet to Disambiguate Word Senses for Text Retrieval. In Proceedings of: *Sixteenth Annual International ACM SIGIR: conference on research developments in information retrieval*, 171 - 180 Pittsburgh, Pennsylvania Association for Computing Machinery.

Waltz, D. L. and J. B. Pollock (1985). "Massively Parallel Parsing." Cognitive Science: a strongly interactive model of natural language interpretation., Vol. 9, : , pp. 51-74.

Weizenbaum, J. (1966). "Eliza: a computer program for the study of natural language communications between man and machine." CACM., Vol. 9, : , pp. 36 - 45.

Wellman, M. P. (1990). "Fundamental Concepts of Qualitative Probabilistic Networks." Artificial Intelligence., Vol. 44, : , pp. 257 - 303.

Wermter, S. (1995). Hybrid Connectionist Natural Language Processing, Chapman & Hall.

Wermter, S. and V. Weber (1996). "Interactive Spoken-Language Processing in a Hybrid Connectionist System." Computer., Vol. 29, (7): , pp. 65.

Wilks, Y. A. (1972). Grammar, Meaning and the Machine Analysis of Natural Language., London, Routledge & Keenan Paul.

Wilks, Y. (1973). "Preference Semantics," Memo AIM-206, Stanford University, California.

Winograd, T. (1972). Understanding Natural Language., New York, Academic Press.

Woods, W. (1972). Progress in Natural Language Understanding: an application to lunar geology. In Proceedings of: *AFIPS*, .

Woods, W. (1978). Semantics and Quantification in Natural Language Question Answering. In (ed. M. Yovitz): Advances in Computers., 17. New York, Academic Press.

Xu, H. (1991). An Efficient Implementation of Belief Function Propagation. In (ed. B. D'Ambrosio, P. Smets and P. Bonissone): Proc 7th Conference on Uncertainty in AI., San Mateo, Morgan Kaufmann: 425 - 432.

Yngve, V. H. (1986). Linguistics as a Science., Indianapolis, Indiana University Press.

Young, S. and G. Bloothooft (1997). Corpus-Based Methods in Language and Speech Processing, Kluwer Academic.

Zadeh, L., A. (1972). "A Rationale for Fuzzy Control." J. Dynamic Systems, Measurement and Control., Vol. 6, (94): , pp. 3 - 4.

Zadeh, L. (1973). "Outline of a new approach to the analysis of complex systems and decisions processes." IEEE Trans. Syst., Man, Cybern., Vol. SMC-3, : , pp. 28-44.

Zernik, V. (1987). Language Acquisition Learning an Hierarchy of Phrases. In Proceedings of: *10th International Conference on Artificial Intelligence,* 125 - 132.

11.0 Appendices.

11.1 Appendix A. (VB code for Modules and components)

11.1.1 Conversation Manager

```
Sub BabyFuz ()
 ReDim ProtoVariables$(2, 0) 'array holding variable strings hypothesised by subtracting  signature patterns from current input stimulus.
  ReDim ProtoFeatures(2, 0) 'array holding signatures hypothesised from correspondence signatures and input
  ReDim ActivatedClusters(2, 0) 'array holding the variable cluster activated by FuzzyResponseVarCorr()

'if last utterance was an hypothesis then detect constant correspondence between new stimuli and last inference
  If LastUtteranceWasAnHypothesis Then
    ThisUtt$ = txtInput.Text
'remove (...) from non-inferential hypotheses
   StripParenthesis ThisUtt$, ReplyHypo$, ReplyUtt$, PartHypo
'detect degree of constant correspondence between hypothesis and current input
   FeatureMatch ReplyHypo$, ReplyUtt$: CalcMatchStrgth: MatchCorrespondence = parse(2, 0)
'if constant correspondence is high between last hypothesis and current stimulus
'then assume current stimulus is a correction of last hypothesis
  If MatchCorrespondence > .66 Or PartHypo = True Then
     lstdb(0).List(lstdb(0).ListCount - 1) = txtInput.Text
```

```
    lstdb(0).AddItem ""
    STM$ = txtInput.Text: Hypothesis$ = "Agooo."
    lstoutput.List(0) = "Agooo.": SelectText: goordinary: LastUtteranceWasAnHypothesis = False
'sleep according to sleep frequency setting
    If InhibitRegualarSleep = False Then
     If SleepFrequency = UtterancesAdded Then UtterancesAdded = 0: SLEEP
    End If
    Exit Sub
  Else 'Else if constant correspondence is low between last hypothesis and current stimulus then assume current stimulus is a continuation of the conversation
      '(and consequently that the last hypothesis was appropriate)
'if last response was not a fact then add variables to appropriate variable base
     If FactualResponse = False Then
      ExtractVariables
'delete utterances which went to make up the assumed correct hypothesis
      lstdb(0).RemoveItem lstdb(0).ListCount - 1
      lstdb(0).RemoveItem lstdb(0).ListCount - 1
      UtterancesAdded = UtterancesAdded - 2
      LastUtteranceWasAnHypothesis = False: Hypothesis$ = "": ClearActivatedSigs
     Else
      FactualResponse = False
     End If
     GoTo FuzzyReply
  End If

Else 'Else if last utterance was not an hypothesis
```

```
'if last utterance was "How should I reply?" then current stimulus is a correction
   If Hypothesis$ = "How should I reply?" Then
    Hypothesis$ = ""
'detect where last text line of textual couplet is and replace it with current stimulus
    If lstdb(0).List(lstdb(0).ListCount - 1) <> "" Then
     lstdb(0).List(lstdb(0).ListCount - 1) = txtInput.Text
     lstdb(0).AddItem ""
    Else
     lstdb(0).List(lstdb(0).ListCount - 2) = txtInput.Text
    End If
    STM$ = txtInput.Text: Hypothesis$ = "Agooo.": lstoutput.List(0) = "Agooo.": txtOutput.Text = "Agooo.": SelectText: goordinary
     If InhibitRegualarSleep = False Then
      If SleepFrequency = UtterancesAdded Then UtterancesAdded = 0: SLEEP 'sleep according to sleep frequency setting
     End If
     Exit Sub
   Else 'Else if last utterance was not "How should I reply?" the current stimulus is a continuation of the conversation
FuzzyReply:
    FactualResponse = False
'compare current stimulus with text couplet list
    CompareTextCouplets Fact$
'if current stimulus matches a text couplet then return reply
    If Fact$ <> "" Then 'if match with textual couplet stimulus
     txtOutput.Text = Fact$ 'reply with textual couplet reply
     Hypothesis$ = Fact$: FactualResponse = True: SelectText: gohappy: Exit Sub
   End If
```

11.1.1.1 StringCorrespondence Evaluator

```
Sub FeatureMatch (ByVal A$, ByVal B$)

  Dim k As Integer 'identifies number of instances of identical match stings in comparitee
  Dim StrtVarA As Integer 'identifies start of each variable in comparitor
  Dim StrtVarB As Integer 'identifies start of each variable in comparitee
  Dim EndVarA As Integer 'identifies end of each variable in comparitor
  Dim EndVarB As Integer 'identifies end of each variable in comparitee
  Dim x$ 'variable elements of A$
  Dim Y$ 'variable elements of B$
  Dim Lngth As Integer 'length of X$
  Dim StrtBit As Integer 'identifies start position of window in comparitor
  Dim Bit$ 'window selected from comparitor
  Dim FoundMatch As Integer 'boolean true if window match between comparitor and comparitee
  Dim ReplyPos As Integer 'identifies position of window match in comparitee
  Dim LegitMatch As Integer 'boolean true if window match between comparitor and comparitee does not compromise heuristics
  Dim BufferA$ 'buffer of A$ used so as not to corrupt A$ during processing
  Dim BufferB$ 'buffer of B$ used so as not to currupt B$ during processing
  Dim Seq As Integer 'loop variable identifying each k
  Dim SeqFound As Integer 'boolean true if sequence of matches complies with all heuristics
  Dim NextSeq As Integer 'loop variable identifying each potential sequence
  Dim LastNum As Integer 'identifies start position of each each element of a sequence
 Dim r As Integer 'matchset variable identifying each sequence set
Dim q As Integer 'matchset variable identifying each sequence element of a set
```

```
Dim ClearArray As Integer 'loop variable identifying each potential sequence already visited for sequencing
Dim Brkts As Integer 'identifies number of character used for a bracketing operation
Dim EndList As Integer 'identifies list number of end of parse list
Dim EachMatchSet As Integer 'identifies postion at end of parse list to place new ambiguities
Dim RB As Integer 'loop variable identifying each parse to have its # and[ removed after processing]


If InStr(1, A$, Chr$(93)) Then
 Bracketed = True
 ConstantsToVariables A$
 ConstantsToVariables B$
Else
 Bracketed = False
 A$ = A$ + "#": B$ = B$ + "#"
End If
If Left$(A$, 1) <> Chr$(93) Then A$ = Chr$(93) + A$: If Right$(A$, 1) <> Chr$(91) Then A$ = A$ + Chr$(91)
If Left$(B$, 1) <> Chr$(93) Then B$ = Chr$(93) + B$: If Right$(B$, 1) <> Chr$(91) Then B$ = B$ + Chr$(91)
parse(0, 0) = A$: parse(1, 0) = B$: k = 0: EachParse = 0
Do Until Len(parse(0, EachParse)) = 0 'FOR EACH PARSE
 Do 'FOR EACH VARIABLE OF EACH PARSE
  StrtVarA = InStr(StrtVarA + 1, parse(0, EachParse), Chr$(93)): EndVarA = InStr(StrtVarA, parse(0, EachParse), Chr$(91))
  StrtVarB = InStr(StrtVarB + 1, parse(1, EachParse), Chr$(93)): EndVarB = InStr(StrtVarB, parse(1, EachParse), Chr$(91))
  x$ = Mid$(parse(0, EachParse), StrtVarA + 1, EndVarA - StrtVarA - 1)
  Y$ = Mid$(parse(1, EachParse), StrtVarB + 1, EndVarB - StrtVarB - 1)
  If Len(x$) > 1 And Len(Y$) > 1 Then
```

```
Lngth = Len(x$)
Do 'LENGTH OF MOVING WINDOW
ReDim MatchPos(-1 To 10, 4): ReDim MatchSet(10, 50, 1): StrtBit = 1
 Do 'MOVING THE WINDOW
  Bit$ = Mid$(x$, StrtBit, Lngth) 'WINDOW
  If InStr(1, Y$, Bit$) <> 0 Then 'IF WINDOW EXISTS IN Y$ THEN FIND IT
   FindMatch Y$, Bit$, StrtBit, FoundMatch, ReplyPos, MatchPos(), k
  End If
  StrtBit = StrtBit + 1
 Loop Until StrtBit + Lngth >= Len(x$) + 2 'END LOOP MOVING WINDOW
      If FoundMatch = True Then
'SHOWARRAY MatchPos(), k
       If k = 1 Then 'IF ONLY ONE MATCH OCCURS FOR A GIVEN LENGTH
        Replace parse(0, EachParse), StrtVarA + MatchPos(0, 0), Len(Bit$), parse(0, EachParse), parse(1, EachParse)
        Replace parse(1, EachParse), StrtVarB + MatchPos(0, 1), Len(Bit$), parse(1, EachParse), parse(0, EachParse)
        LegitMatch = True
       Else 'IF MORE THAN ONE MATCH OCCURS FOR A GIVEN LENGTH
        BufferA$ = parse(0, EachParse): BufferB$ = parse(1, EachParse)
        For Seq = 0 To k - 1 'FOR EACH START MATCH
         If MatchPos(Seq, 3) <> True Then
         Do 'LOOP FOR EACH SEQUENCE FOR EACH START MATCH
         SeqFound = False
         For NextSeq = Seq + 1 To k - 1 'FOR EACH SEQUENCE OF EACH START MATCH
          If MatchPos(NextSeq, 1) > MatchPos(Seq, 1) + Len(Bit$) - 1 Then 'IF (NEXT IS LARGER THAN END OF FIRST FOR $2
           If MatchPos(NextSeq, 0) > MatchPos(Seq, 0) + Len(Bit$) - 1 Then 'NEXT IS LARGER THAN END OF FIRST FOR $1
```

```
    If MatchPos(NextSeq, 1) > LastNum + Len(Bit$) - 1 And MatchPos(NextSeq, 0) > LastNum1 + Len(Bit$) - 1 Then 'LARGER THAN END OF LARGEST
     If MatchPos(NextSeq, 2) <> True And MatchPos(LastNumRec, 4) <> True Then 'NOT VISITED IN $2
      MatchSet(0, r, 0) = MatchPos(Seq, 0): MatchSet(0, r, 1) = MatchPos(Seq, 1)
      MatchSet(q + 1, r, 0) = MatchPos(NextSeq, 0): MatchSet(q + 1, r, 1) = MatchPos(NextSeq, 1)
      MatchPos(NextSeq, 2) = True: MatchPos(NextSeq, 3) = True: MatchPos(Seq, 3) = True
      q = q + 1: SeqFound = True: LastNumRec = LastNum1: LastNum1 = MatchPos(NextSeq, 0)
      LastNum = MatchPos(NextSeq, 1): LegitMatch = True
     End If
    End If
   End If
End If
'IF SEQUENCE IN $2 IS LEGIT BUT DOES NOT FORM PART OF A SEQUENCE BECAUSE OF CONSTRAINTS IN $1 THEN SET FLAG AND DELETE VISITED THIS
'SEQUENCE IN SUBSEQUENT ELEMENTS OF $2
          If MatchPos(NextSeq, 0) > LastNumRec + Len(Bit$) - 1 = False And MatchPos(NextSeq, 4) = True Then
           For ClearArray = NextSeq + 1 To k - 1
            MatchPos(ClearArray, 2) = False
           Next ClearArray
          End If
          If MatchPos(NextSeq, 0) > LastNumRec + Len(Bit$) - 1 = False Then
           MatchPos(LastNumRec, 4) = True: MatchPos(NextSeq, 4) = True '$1 constraint on an otherwise legit sequence
          End If
          LastNumRec = 0
         Next NextSeq 'LOOP NEXT MATCH AS CANDIDATE FOR SEQUENCE
        If SeqFound = True Then r = r + 1: q = 0: LastNum = 0: LastNum1 = 0
       Loop Until SeqFound = False 'END LOOP FOR THIS SEQUENCE OF GIVEN START MATCH
```

```
      For ClearArray = 0 To k - 1 'CLEAR VISITED FOR THIS SEQUENCE FLAGS
       MatchPos(ClearArray, 2) = False: MatchPos(ClearArray, 4) = False
      Next ClearArray
    End If
  Next Seq 'LOOP NEXT SEQUENCE START
End If
For ClearArray = 0 To k - 1 'RECORD NON-SEQUENTIAL MATCHES
  If Len(MatchPos(ClearArray, 3)) = 0 Then
    MatchSet(0, r, 0) = MatchPos(ClearArray, 0)
    MatchSet(0, r, 1) = MatchPos(ClearArray, 1)
     r = r + 1: LegitMatch = True
    End If
    MatchPos(ClearArray, 3) = False 'CLEAR NEVER BEEN PART OF A SEQUENCE
   Next ClearArray
'SHOWARRAY MatchSet(), r
    If k > 1 Then 'REPLACE AMBIGUOUS MATCHES
'REPLACE FIRST AMBIGUITY IN CURRENT PARSE STRING
     Do
      Replace parse(0, EachParse), StrtVarA + MatchSet(q, 0, 0) + Brkts, Len(Bit$), parse(0, EachParse), parse(1, EachParse)
      Replace parse(1, EachParse), StrtVarB + MatchSet(q, 0, 1) + Brkts, Len(Bit$), parse(1, EachParse), parse(0, EachParse)
      q = q + 1: Brkts = Brkts + 2
     Loop Until Len(MatchSet(q, 0, 0)) = 0
     Brkts = 0: q = 0
```

```
'FIND END OF PARSE LIST
        Do
         If Len(parse(0, EndList)) <> 0 Then EndList = EndList + 1
        Loop Until Len(parse(0, EndList)) = 0
'REPLACE SUBSEQUENT AMBIGUITIES AT END OF PARSE LIST
        For EachMatchSet = EndList To EndList + (r - 2)
         parse(0, EachMatchSet) = BufferA$: parse(1, EachMatchSet) = BufferB$
          Do
           Replace parse(0, EachMatchSet), StrtVarA + MatchSet(q, EachMatchSet - EndList + 1, 0) + Brkts, Len(Bit$), parse(0, EachMatchSet), parse(1, EachMatchSet)
           Replace parse(1, EachMatchSet), StrtVarB + MatchSet(q, EachMatchSet - EndList + 1, 1) + Brkts, Len(Bit$), parse(1, EachMatchSet), parse(0, EachMatchSet)
           q = q + 1: Brkts = Brkts + 2
          Loop Until Len(MatchSet(q, EachMatchSet, 1)) = 0
          Brkts = 0: q = 0
         Next EachMatchSet
        End If
        Brkts = 0: Erase MatchPos: Erase MatchSet: k = 0: Lngth = Lngth - 1: r = 0: EndList = 0: Exit Do
       End If
       Lngth = Lngth - 1
      Loop Until Lngth < 2 'LOOP MOVING WINDOWS TILL LENGTH < 2
      If LegitMatch = True Then StrtVarA = 0: StrtVarB = 0
      FoundMatch = False: LegitMatch = False: StrtBit = 1
    End If
  Loop Until InStr(StrtVarA + 1, parse(0, EachParse), Chr$(93)) = 0 Or InStr(StrtVarB + 1, parse(1, EachParse), Chr$(93)) = 0 'LOOP FOR EACH VARIABLE
  EachParse = EachParse + 1: StrtVarA = 0: StrtVarB = 0
Loop 'LOOP FOR EACH PARSE
```

```
For RB = 0 To EachParse - 1 'REMOVE START/END BRACKETS
  RemoveHatch parse(0, RB)
  RemoveHatch parse(1, RB)
Next RB

If Bracketed = True Then
  For RB = 0 To EachParse - 1
    RemoveEmptyConstants parse(0, RB)
    RemoveEmptyConstants parse(1, RB)
  Next RB
End If

End Sub
```

11.1.1.2 StimulusTextCouplet Comparitor

```
Sub CompareTextCouplets (Fact$)
'compares input stimulus with existing textual coupet stimuli, if an exact match is found then outputs associated textual couplet response.

 For EachTextStim = 0 To lstdb(0).ListCount - 1 Step 3
  If Utterance$ = lstdb(0).List(EachTextStim) Then
   Fact$ = lstdb(0).List(EachTextStim + 1)
   Exit Sub
  End If
 Next EachTextStim

End Sub
```

11.1.1.3 Variable Extractor

```
Sub ExtractVariables ()
'sub routine detects variables which exist when signatures are subtracted from them and stores
'them in a list such that if the list is placed directly underneath their corresponding pattern list they appear in the same positions

If lstdb(0).ListCount <> 0 Then
   ReDim C$(-1 To 0) 'contains constants from patterns so they can be detected in utternaces
   ReDim VarStim$(3, -1 To 0) 'contains variables detected from utterance stimuli
   ReDim VarResp$(3, -1 To 0) 'contains variables detected from utterance responses
   ReDim DeleteUtterance(lstdb(0).ListCount - 1) 'utterance list number flag, true if utterance forms both a constant correspondence and an
                                                 'hypthosesisabe feature couplet with a pattern. Flag is used to delete utterances at end.

'for each signature base
   For EachBase = 23 to 29
    TargetBase = -1
'for each signature pair in the current base
    Do Until EachPattern >= lstdb(EachBase).ListCount - 1
     NoCon1 = False: NoCon2 = False: NewPat = True: FirstAdd = True: NewPatternFormed = False
     ConCorrFound = False 'true if a set of textual couplets has produced and constant correspondence
'get stimulus and response pattern
    PatStim$ = lstdb(EachBase).List(EachPattern)
    PatResp$ = lstdb(EachBase).List(EachPattern + 1)
```

```
        If PatStim$ <> "" And PatResp$ <> "" Then
 'transfer signature couplet to feature base
          lstdb(EachBase - 22).AddItem PatStim$
          lstdb(EachBase - 22).AddItem PatResp$
          lstdb(EachBase - 22).AddItem ""
'for each textual couplet
        For EachUtterance = 0 To lstdb(0).ListCount - 1 Step 3
          UttStim$ = lstdb(0).List(EachUtterance)

'detect whether constant equivalence exists between PatStim$ and UttStim$
            ConstantEquivalence PatStim$, UttStim$, ConCorr, NoCon1
 'if utterance constant correspondence is true
          If ConCorr = True Then
            UttResp$ = lstdb(0).List(EachUtterance + 1)
'detect whether constant correspondence exists between PatStim$ and UttStim$
            ConstantEquivalence PatResp$, UttResp$, ConCorr, NoCon2
          End If

'if constant correspondence exists between stimulus + response patterns & utterances
          If ConCorr = True Then 'And NoCon1 = False And NoCon2 = False Then
            ConCorrFound = True
'mine for variables in stimulus signature
            VarMine UttStim$, PatStim$, VarStim$(), VarResp$(), NumVar, 0, LastNumVar
            LastNumVar = NumVar: NumVar = 0
```

```
'mine for variables in corresponding response signature
        VarMine UttResp$, PatResp$, VarResp$(), VarStim$(), NumVar, 1, LastNumVar
        DeleteUtterance(EachUtterance) = True
'if new pattern, then form buffer of stimulus and reponse pattern
        If NewPat Then
         StimTemp$ = lstdb(EachBase).List(EachPattern)
         RespTemp$ = lstdb(EachBase).List(EachPattern + 1)
         NewPat = False
        End If
        StimBuf$ = StimTemp$: RespBuf$ = RespTemp$

'construct stimulus feature in buffer
        Do
         StrtVar = InStr(StrtVar + 1, StimBuf$, "]")
         EndVar = InStr(StrtVar, StimBuf$, "[")
         StimBuf$ = Left$(StimBuf$, StrtVar - 1) + VarStim$(1, EachVarSlot) + Right$(StimBuf$, Len(StimBuf$) - EndVar)
         StrtVar = StrtVar + Len(VarStim$(1, EachVarSlot)): EachVarSlot = EachVarSlot + 1
        Loop Until InStr(StrtVar, StimBuf$, "]") = 0
        StrtVar = 0: EndVar = 0

'construct corresponding response featuer in buffer
        Do
         StrtVar = InStr(StrtVar + 1, RespBuf$, "]")
         EndVar = InStr(StrtVar, RespBuf$, "[")
         RespBuf$ = Left$(RespBuf$, StrtVar - 1) + VarStim$(1, EachVarSlot) + Right$(RespBuf$, Len(RespBuf$) - EndVar)
```

```
    StrtVar = StrtVar + Len(VarStim$(1, EachVarSlot)): EachVarSlot = EachVarSlot + 1
  Loop Until InStr(StrtVar, RespBuf$, "]") = 0
  EachVarSlot = 0: StrtVar = 0: EndVar = 0

'check whether pattern already exists in current feature base
  ExistPat = False
  For EachPat = 0 To lstdb(EachBase - 22).ListCount - 1 Step 3
'if constructed stimulus and response pattern are same as already in current feature base then set ExistsPat to true and increment EachPat: exit loop
    If StimBuf$ = lstdb(EachBase - 22).List(EachPat) And RespBuf$ = lstdb(EachBase - 22).List(EachPat + 1) Then
      ExistPat = True: EachPat = EachPat + 3: Exit For
    End If
  Next EachPat
  EachPat = EachPat - 3
'if constructed feature couplet exist in feature base
  If ExistPat Then
'construct variable cluster and check whether it already exists in variable base
    SBuf$ = SBuf$ & "/"
    For EachVar = 0 To LastNumVar
      SBuf$ = SBuf$ & VarStim$(2, EachVar)
    Next EachVar
    RBuf$ = RBuf$ & "/"
    For EachVar = LastNumVar + 1 To LastNumVar + NumVar + 1
      RBuf$ = RBuf$ & VarStim$(2, EachVar)
    Next EachVar
'if stimulus and response do not contain variable cluster to be added
```

```
            If InStr(1, lstdb(EachBase - 14).List(EachPat), SBuf$) And InStr(1, lstdb(EachBase - 14).List(EachPat + 1), RBuf$) Then
            Else
'add variables to appropriate variable list
              lstdb(EachBase - 14).List(EachPat) = lstdb(EachBase - 14).List(EachPat) + SBuf$
              lstdb(EachBase - 14).List(EachPat) = lstdb(EachBase - 14).List(EachPat) + "~"
              lstdb(EachBase - 14).List(EachPat + 1) = lstdb(EachBase - 14).List(EachPat + 1) + RBuf$
              lstdb(EachBase - 14).List(EachPat + 1) = lstdb(EachBase - 14).List(EachPat + 1) + "~"
            End If
            SBuf$ = "": RBuf$ = ""
'if constructed stimulus and response patterns are not already in current signature base
          Else 'if this is the first time that a signature is being added for this set of textual coupets then replace the existing feature
               'couplet in current feature base and add variables

           If FirstAdd = True Then
             NewPatternFormed = True
'add feature couplet to current feature base
             lstdb(EachBase - 22).List(EachPat) = StimBuf$
             lstdb(EachBase - 22).List(EachPat + 1) = RespBuf$
'if this signature comes from an earlier pattern then make room on feature base
             If lstdb(EachBase).List(EachPat) <> PatStim$ Then
'add space at end of feature base list
              For X = 0 To 2
               lstdb(EachBase).AddItem ""
             Next X
'move signature base list down three lines
```

```
            For X = lstdb(EachBase).ListCount - 1 To EachPat + 4 Step -1
             lstdb(EachBase).List(X - 1) = lstdb(EachBase).List(X - 4)
             lstdb(EachBase).List(X - 4) = ""
            Next X
           End If
           If lstdb(EachBase - 14).ListCount - 1 < EachPat + 2 Then
            Do Until lstdb(EachBase - 14).ListCount - 1 = EachPat + 2
             lstdb(EachBase - 14).AddItem ""
            Loop
           End If
'add varibles
           lstdb(EachBase - 14).List(EachPat) = lstdb(EachBase - 14).List(EachPat) + "/"
           For EachVar = 0 To LastNumVar
            lstdb(EachBase - 14).List(EachPat) = lstdb(EachBase - 14).List(EachPat) + VarStim$(2, EachVar)
           Next EachVar
           lstdb(EachBase - 14).List(EachPat) = lstdb(EachBase - 14).List(EachPat) + "~"

           lstdb(EachBase - 14).List(EachPat + 1) = lstdb(EachBase - 14).List(EachPat + 1) + "/"
           For EachVar = LastNumVar + 1 To LastNumVar + NumVar + 1
            lstdb(EachBase - 14).List(EachPat + 1) = lstdb(EachBase - 14).List(EachPat + 1) + VarStim$(2, EachVar)
           Next EachVar
           lstdb(EachBase - 14).List(EachPat + 1) = lstdb(EachBase - 14).List(EachPat + 1) + "~"
           FirstAdd = False
'if this is not the first time that a feature couplet is being added for this set of textual coupets then add the Feature couplet to the end ofcurrent
'Feature couplet base and add variables
```

```
        Else
          NewPatternFormed = True': EachPattern = EachPattern + 3
'add feature couplet to current feature base
          lstdb(EachBase - 22).AddItem StimBuf$
          lstdb(EachBase - 22).AddItem RespBuf$
          lstdb(EachBase - 22).AddItem ""
'add room in current variable base to accept variables
          lstdb(EachBase - 14).AddItem ""
          lstdb(EachBase - 14).AddItem ""
          lstdb(EachBase - 14).AddItem ""
'add space in signature base for multi formation of features
          lstdb(EachBase).AddItem ""
          lstdb(EachBase).AddItem ""
          lstdb(EachBase).AddItem ""
'move feature base list down three lines
          For X = lstdb(EachBase).ListCount - 1 To EachPat + 6 Step -1
           lstdb(EachBase).List(X - 1) = lstdb(EachBase).List(X - 4)
           lstdb(EachBase).List(X - 4) = ""
          Next X
'add varibles
          lstdb(EachBase - 14).List(EachAddition) = lstdb(EachBase - 14).List(EachAddition) + "/"
          For EachVar = 0 To LastNumVar
           lstdb(EachBase - 14).List(EachAddition) = lstdb(EachBase - 14).List(EachAddition) + VarStim$(2, EachVar)
         Next EachVar
          lstdb(EachBase - 14).List(EachAddition) = lstdb(EachBase - 14).List(EachAddition) + "~"
```

```
    lstdb(EachBase - 14).List(EachAddition + 1) = lstdb(EachBase - 14).List(EachAddition + 1) + "/"
   For EachVar = LastNumVar + 1 To LastNumVar + NumVar + 1
    lstdb(EachBase - 14).List(EachAddition + 1) = lstdb(EachBase - 14).List(EachAddition + 1) + VarStim$(2, EachVar)
   Next EachVar
   lstdb(EachBase - 14).List(EachAddition + 1) = lstdb(EachBase - 14).List(EachAddition + 1) + "~"
  End If
  If NewPatternFormed = True Then
   EachAddition = EachAddition + 3
   NewPatternFormed = False
  End If
        End If
        ReDim VarResp$(3, -1 To 0): ReDim VarStim$(3, -1 To 0): NumVar = 0
       End If
      Next EachUtterance
'else if there was no constant correspondence with any utterance or the variables generated went to another signature then delete candidate
'feature from current signature base
      If lstdb(EachBase - 14).List(lstdb(EachBase - 22).ListCount - 2) = "" Then
       lstdb(EachBase - 22).RemoveItem lstdb(EachBase - 22).ListCount - 1
       lstdb(EachBase - 22).RemoveItem lstdb(EachBase - 22).ListCount - 1
       lstdb(EachBase - 22).RemoveItem lstdb(EachBase - 22).ListCount - 1
      End If
    End If 'If PatStim$ <> "" And PatResp$ <> ""
    EachPattern = EachPattern + 3
   Loop 'each pattern
```

```
    EachPattern = 0: EachAddition = 0
  Next EachBase

  If DeleteCouplets = True Then
'delete utterances used to generate variables
    For deleteable = lstdb(0).ListCount - 3 To 0 Step -3
      If DeleteUtterance(deleteable) = True Then
        lstdb(0).RemoveItem deleteable
        lstdb(0).RemoveItem deleteable
        lstdb(0).RemoveItem deleteable
      End If
    Next deleteable
  End If
 End If

End Sub
```

11.1.2 Pattern Extractor.

11.1.2.1 Extractor Manager

```
Sub SLEEP ()

 If lstdb(0).ListCount = 0 Then Exit Sub
 gosleepy
 Gauge1.Visible = True
  DoEvents
  FirstPass = True: SignatureDetected = False

'text couplets to buffer 3
   For EachUtterance = 0 To lstdb(0).ListCount - 1
    lstbuf(3).List(EachUtterance) = lstdb(0).List(EachUtterance)
  Next EachUtterance

 Do
'text couplets to buffer 1
  For EachUtterance = 0 To lstdb(0).ListCount - 1
   lstbuf(1).List(EachUtterance) = lstdb(0).List(EachUtterance)
  Next EachUtterance
  lstdb(0).Clear
```

```
If PreselectOn = True Then
    PreselectControl 'from Buffer1 to Buffer2
'if no more preselection can take place then exit program
    If lstbuf(2).ListCount = 0 Then Exit Sub
    Gauge1.Value = Gauge1.Value + 5
'buffer 1 to text couplets provided Buffer1 contains a pair of text couplets
    If lstbuf(1).ListCount - 1 > 3 Then
      For EachUtterance = 0 To lstbuf(1).ListCount - 1
        lstdb(0).List(EachUtterance) = lstbuf(1).List(EachUtterance)
      Next EachUtterance
      End If
      lstbuf(1).Clear
 'buffer 2 to buffer 1
      For EachUtterance = 0 To lstbuf(2).ListCount - 1
        lstbuf(1).List(EachUtterance) = lstbuf(2).List(EachUtterance)
      Next EachUtterance
      lstbuf(2).Clear
    End If
 DetectPatterns MatchStrength  'from Buffer1 to Buffer2
    Gauge1.Value = Gauge1.Value + 5
'if the match is significant then process signatures
   If MatchStrength >= .1 Then
    SignatureDetected = True: lstbuf(1).Clear
'get list number of first new entry to lstdb(1)
    If FirstPass = True Then
```

```
        FirstEntry = lstdb(1).ListCount
        FirstPass = False
      End If
'from buffer2 to feature couplets + space for variables (provided they don't already exist)
      DuplicateSignature = False
      For EachNewSignature = 0 To lstbuf(2).ListCount - 1 Step 3
       For EachOldSignature = lstdb(23).ListCount - 3 To 0 Step -3
        If lstbuf(2).List(EachNewSignature) = lstdb(23).List(EachOldSignature) Then
         If lstbuf(2).List(EachNewSignature + 1) = lstdb(23).List(EachOldSignature + 1) Then
         DuplicateSignature = True
         Exit For
         End If
        End If
       Next EachOldSignature
       If DuplicateSignature = False Then
'add signature pattern to signature feature base
        lstdb(23).AddItem lstbuf(2).List(EachNewSignature)
        lstdb(23).AddItem lstbuf(2).List(EachNewSignature + 1)
        lstdb(23).AddItem ""
'add space for variables
        lstdb(9).AddItem ""
        lstdb(9).AddItem ""
        lstdb(9).AddItem ""
      End If
      DuplicateSignature = False
```

```
    Next EachNewSignature
    lstbuf(2).Clear
   End If
   Gauge1.Value = Gauge1.Value + 5
  Loop Until lstdb(0).List(3) = ""
'from buffer3 to text couplets
  For EachUtterance = 0 To lstbuf(3).ListCount - 1
   lstdb(0).AddItem lstbuf(3).List(EachUtterance)
  Next EachUtterance
  lstbuf(3).Clear
   Gauge1.Value = Gauge1.Value + 5
   If SignatureDetected = False Then GoTo ExVar
   lstNum = 1: LstNum1 = 23
'for each meta base
   Do
    FirstPass = True
'feature couplets to buffer1
    For EachUtterance = FirstEntry To lstdb(LstNum1).ListCount - 1
     lstbuf(1).List(EachUtterance - FirstEntry) = lstdb(LstNum1).List(EachUtterance)
    Next EachUtterance
'recurse 'preselect' and 'pattern detect' until lstbuf(1) contains less than two feature couplets
   Do
    PreselectControl 'from Buffer1 to Buffer2
    Gauge1.Value = Gauge1.Value + 5
```

```
'if no more preselection can take place then exit program
    If lstbuf(2).ListCount = 0 Then GoTo ExVar
'buffer1 to over
    For EachUtterance = 0 To lstbuf(1).ListCount - 1
      lstbuf(0).List(EachUtterance) = lstbuf(1).List(EachUtterance)
    Next EachUtterance
    lstbuf(1).Clear
'buffer2 to buffer1
    For EachUtterance = 0 To lstbuf(2).ListCount - 1
      lstbuf(1).List(EachUtterance) = lstbuf(2).List(EachUtterance)
    Next EachUtterance
    lstbuf(2).Clear

    DetectPatterns MatchStrength 'from Buffer1 to Buffer2
    Gauge1.Value = Gauge1.Value + 5
'if the match is significant then process meta signatures
    If MatchStrength >= .1 Then
'over to buffer1
     For EachUtterance = 0 To lstbuf(0).ListCount - 1
       lstbuf(1).List(EachUtterance) = lstbuf(0).List(EachUtterance)
     Next EachUtterance
     lstbuf(0).Clear
'from buffer2 to meta couplets + space for variables
'get list number of first new entry to lstdb(LstNum + 1)
    If FirstPass = True Then
```

```
        FirstEntry = lstdb(lstNum + 1).ListCount
        FirstPass = False
      End If

'no need to check for duplicates as they don't exist in the root list

'add metafeatures to metafeature bases and metasignature bases
      For MetaFeature = 0 To lstbuf(2).ListCount - 1
        lstdb(LstNum1 + 1).AddItem lstbuf(2).List(MetaFeature)
        lstdb(lstNum + 9).AddItem ""
        Next MetaFeature
        lstbuf(2).Clear
      End If
    Loop Until lstbuf(1).List(3) = ""
    lstNum = lstNum + 1: LstNum1 = LstNum1 + 1
  'loop until metabase being slept on has less than two meta feature couplets
  Loop Until lstdb(LstNum1).ListCount = 2
  'clear buffers
ExVar: lstbuf(0).Clear
 lstbuf(1).Clear
 lstbuf(2).Clear
 Gauge1.Value = Gauge1.Value + 5
 If SignatureDetected = True Then
  ExtractVariables
End If
```

```
 goordinary
 Gauge1.Value = 100
 For xx = 0 To 10000
 Next xx
 Gauge1.Visible = False
 Gauge1.Value = 0

If PrintAnalysisOn = True Then
  PrintAnalysis
End If

End Sub
```

11.1.2.2 DominantFeatureCouplet Pre-selector

```
Sub Preselect (pos(), Score(), EachMatch, MaxCands)
'inputs a set of textual couplets and returns a list of the position of those which
'contain the longest most frequent constants- (determined by character length * occurrance)

   For EachStim = 0 To lstbuf(1).ListCount - 1 Step 3
'get stimulus element
     X$ = lstbuf(1).List(EachStim)
     Lngth = Len(X$)
     Do 'LENGTH OF MOVING WINDOW
     StrtBit = 1
      Do 'MOVING THE WINDOW
       Bit$ = Mid$(X$, StrtBit, Lngth) 'WINDOW
       For EachStimComp = EachStim + 3 To lstbuf(1).ListCount - 1 Step 3
        Y$ = lstbuf(1).List(EachStimComp)
'count number of constants in X$ and Y$
        CountConsts X$, Y$, SameConsts
        If SameConsts = True Then
         If Len(Bit$) <= Len(Y$) Then
          If InStr(1, Y$, Bit$) <> 0 Then 'IF MATCH EXISTS IN Y$ THEN FIND IT
'if current match = target pattern
           If Bit$ = TargetPattern Then
            If Cands + 1 > MaxCands Then
```

```
                ReDim Preserve pos(10, Cands + 1)
              End If
'store the list number of current comparitor stimulus
              pos(EachMatch, Cands) = EachStim
'store the list number of each comparison stimuli
              pos(EachMatch, Cands + 1) = EachStimComp
'score how many candidate comparison stimuli match with comparitor
              Score(EachMatch) = Score(EachMatch) + Len(Bit$)
'detect largest score
              Cands = Cands + 2: If Cands > MaxCands Then MaxCands = Cands
            End If
'if match length is greater than longest so far
            If Len(Bit$) > TgtLngth Then
'reset data
              TgtLngth = Len(Bit$): Cands = 0: EachMatch = EachMatch + 1: TargetPattern = Bit$
              pos(EachMatch, Cands) = EachStim
              pos(EachMatch, Cands + 1) = EachStimComp
              Score(EachMatch) = Len(Bit$)
              Cands = Cands + 2: FoundMatch = True
              If MaxCands = 0 Then MaxCands = Cands
            End If
          End If
        End If
      End If
    Next EachStimComp
```

```
    If FoundMatch = True Then Exit Do
    StrtBit = StrtBit + 1
  Loop Until StrtBit + Lngth >= Len(X$) + 2 'END LOOP MOVING WINDOW
  If FoundMatch = True Then FoundMatch = False: Exit Do
  Lngth = Lngth - 1
 Loop Until Lngth < TgtLngth 'LOOP MOVING WINDOWS TILL LENGTH < Target length
Next EachStim

End Sub
```

11.1.2.3 DominantFeatureCouplet Detector.

```
Sub DetectPatterns (MatchStrength) 'background without variable numbers

 ReDim TrgtX(2, 0)'contains list number of target textual couplets with associated target comparitors and used in search flag
 ReDim TrgtStimPatterns(0) 'contains numbered stimulus patterns of target text couplet matches
 ReDim TrgtRespPatterns(0) 'contains numbered response patterns of target text couplet matches
 ReDim StimParse(2, 20) 'stores parse() so that analysis remains after response feature match
 ReDim OrderedX(0) 'contains ordered version of array

   Do

  'do for each new txtcplt to be compared with the others
     Do
       Gauge1.Value = Gauge1.Value + (lstbuf(1).ListCount - 1) / 20
  'exit background processing when no more feature couplets are to be made
       If lstbuf(1).List(3) = "" Then Exit Do
       NextComparitor = NextComparitor + 3
 'for each old textual couplet stimulus
       For X = lstbuf(1).ListCount - NextComparitor - 3 To 0 Step -3
 'collect current comparitor and old stimulus element of textual couplet
       NewStim$ = lstbuf(1).List(lstbuf(1).ListCount - NextComparitor)
       OldStim$ = lstbuf(1).List(X)
 'detect whether string contains brackets and set flag
```

```
    If InStr(1, NewStim$, "[") <> 0 Then
      Bracketed = True
    Else
      Bracketed = False
    End If
    StripVarNos NewStim$, NewStim$: StripVarNos OldStim$, OldStim$
    Erase parse: Erase StimParse: ReDim StimParse(2, 20)
'feature match current comparitor and old stimulus element of textual couplet
    FeatureMatch NewStim$, OldStim$
'calculate stimulus match strength
    CalcMStrngth Strngth
    StimMatch = Strngth
'store the number of resulting parses
'transfer parse() to StimParse() so that parse can be used again and compared with old
     For StoreParse = 0 To EachParse - 1
       StimParse(0, StoreParse) = parse(0, StoreParse)
       StimParse(1, StoreParse) = parse(1, StoreParse)
     Next StoreParse
     YParse = EachParse - 1: EachParse = 0
     Do 'for each parse of stimulus
      NewStimPat$ = StimParse(0, Y): OldStim$ = StimParse(1, Y)
      StripVarNos NewStimPat$, NewStimPat$
      StripVarNos OldStim$, OldStim$
     If NewStimPat$ = OldStim$ Then 'if constant correspondence exists between new and old stimuli
      If Y = 0 Then
```

```
        Erase parse
        NewResp$ = lstbuf(1).List(lstbuf(1).ListCount - NextComparitor + 1)
        OldResp$ = lstbuf(1).List(X + 1)
        StripVarNos NewResp$, NewResp$
        StripVarNos OldResp$, OldResp$
        FeatureMatch NewResp$, OldResp$
'calculate response match strength
        CalcMStrngth Strngth
        RespMatch = Strngth: Strngth = 0
'detect greatest match strength
        If StimMatch > RespMatch Then
          MatchStrength = StimMatch
        Else
          MatchStrength = RespMatch
        End If
        ZParse = EachParse - 1: EachParse = 0
      End If
      Do 'for each parse of response
        NewResp$ = parse(0, Z): OldResp$ = parse(1, Z)
        StripVarNos NewResp$, NewResp$
        StripVarNos OldResp$, OldResp$
        If NewResp$ = OldResp$ Then 'if constant correspondence exists between new and old responses
          FoundMatch = True
          Len1 = Len(NewStimPat$)
          Len2 = Len(NewResp$)
```

```
Length = Len1 + Len2
If Length = TargetLngth Then
 ReDim Preserve TrgtX(2, NumOfTrgts + 1)
 TrgtX(0, NumOfTrgts + 1) = lstbuf(1).ListCount - NextComparitor
 TrgtX(1, NumOfTrgts + 1) = X
 ReDim Preserve TrgtRespPatterns(NumOfTrgts + 1)
 ReDim Preserve TrgtStimPatterns(NumOfTrgts + 1)
 TrgtRespPatterns(NumOfTrgts + 1) = NewResp$
 TrgtStimPatterns(NumOfTrgts + 1) = NewStimPat$
 NumOfTrgts = NumOfTrgts + 1
            End If
            If Length > TargetLngth Then
'if constant length > the current longest then store vars(), NewStimPat, NewResp, v0, v1, v2, v3
             TargetLngth = Length: ReDim TrgtStimPattern(0): ReDim TrgtRespPatterns(0)
             ReDim TrgtX(2, 0): NumOfTrgts = 0
             TrgtStimPatterns(0) = NewStimPat$: TrgtRespPatterns(0) = NewResp$
             TrgtX(1, 0) = X
             TrgtX(0, 0) = lstbuf(1).ListCount - NextComparitor
            End If
           End If
           Z = Z + 1: LngstCon = 0: LngstConR = 0
          Loop Until Z = ZParse + 1
         End If
         Y = Y + 1: Z = 0
        Loop Until Y = YParse + 1
```

```
        Y = 0
      Next X
    Loop Until lstbuf(1).ListCount - NextComparitor = 3
    If lstbuf(1).List(3) = "" Then Exit Do
    For comadd = NumOfTrgts + 1 To NumOfTrgts * 2 + 1
     ReDim Preserve TrgtX(2, comadd)
     TrgtX(1, comadd) = TrgtX(0, dd)
     dd = dd + 1
    Next comadd
   dd = 0
'order trgtx and trgtnextcomparitors
    OrderArray NumOfTrgts * 2 + 1, TrgtX(), OrderedX()
 'remove duplicates in ordered trgtx
    RemoveArrayDuplicates1d OrderedX(), NumOfTrgts * 2 + 1, NumRmvd
    If FoundMatch = True Then
'for the number of target patterns found update ftrcplt and var bases
     For EachChosen = 0 To NumOfTrgts
      UpdateBases TrgtStimPatterns(EachChosen), TrgtRespPatterns(EachChosen)
     Next EachChosen
'delete text couplets which formed new signatures
    For EachChosen = 0 To NumOfTrgts * 2 + 1 - NumRmvd
     lstbuf(1).RemoveItem OrderedX(EachChosen)
     lstbuf(1).RemoveItem OrderedX(EachChosen)
     lstbuf(1).RemoveItem OrderedX(EachChosen)
    Next EachChosen
```

```
  Else
    Exit Do
  End If
  TargetLngth = 0: NumOfTrgts = 0: FoundMatch = False: TargetCon$ = "": TargetLngstCon = 0
  EachChosen = 0: NumRmvd = 0: NextComparitor = 0: ReDim OrderedX(0)
  ReDim TrgtRespPatterns(0): ReDim TrgtStimPatterns(0): LngstCon = 0: LngstConR = 0
 Loop
 Y = 0

End Sub
```

11.1.2.4 Feature Detector.

```
Sub FeatureMatch (ByVal A$, ByVal B$)

 Dim k As Integer 'identifies number of instances of identical match stings in comparitee
 Dim StrtVarA As Integer 'identifies start of each variable in comparitor
 Dim StrtVarB As Integer 'identifies start of each variable in comparitee
 Dim EndVarA As Integer 'identifies end of each variable in comparitor
 Dim EndVarB As Integer 'identifies end of each variable in comparitee
 Dim x$ 'variable elements of A$
 Dim Y$ 'variable elements of B$
 Dim Lngth As Integer 'length of X$
 Dim StrtBit As Integer 'identifies start position of window in comparitor
 Dim Bit$ 'window selected from comparitor
 Dim FoundMatch As Integer 'boolean true if window match between comparitor and comparitee
 Dim ReplyPos As Integer 'identifies position of window match in comparitee
 Dim LegitMatch As Integer 'boolean true if window match between comparitor and comparitee
 Dim BufferA$ 'buffer of A$ used so as not to corrupt A$ during processing
 Dim BufferB$ 'buffer of B$ used so as not to currupt B$ during processing
 Dim Seq As Integer 'loop variable identifying each k
 Dim SeqFound As Integer 'boolean true if sequence of matches complies with all heuristics
 Dim NextSeq As Integer 'loop variable identifying each potential sequence
 Dim LastNum As Integer 'identifies start position of each each element of a sequence
 Dim r As Integer 'matchset variable identifying each sequence set
```

```
Dim q As Integer 'matchset variable identifying each sequence element of a set
Dim ClearArray As Integer 'loop variable identifying each potential sequence already visited for sequencing
Dim Brkts As Integer 'identifies number of character used for a bracketing operation
Dim EndList As Integer 'identifies list number of end of parse list
Dim EachMatchSet As Integer 'identifies postion at end of parse list to place new ambiguities
Dim RB As Integer 'loop variable identifying each parse to have its # and[ removed after processing


If InStr(1, A$, Chr$(93)) Then
  Bracketed = True
  ConstantsToVariables A$
  ConstantsToVariables B$
Else
  Bracketed = False
  A$ = A$ + "#": B$ = B$ + "#"
End If
If Left$(A$, 1) <> Chr$(93) Then A$ = Chr$(93) + A$: If Right$(A$, 1) <> Chr$(91) Then A$ = A$ + Chr$(91)
If Left$(B$, 1) <> Chr$(93) Then B$ = Chr$(93) + B$: If Right$(B$, 1) <> Chr$(91) Then B$ = B$ + Chr$(91)
parse(0, 0) = A$: parse(1, 0) = B$: k = 0: EachParse = 0
Do Until Len(parse(0, EachParse)) = 0 'FOR EACH PARSE
 Do 'FOR EACH VARIABLE OF EACH PARSE
  StrtVarA = InStr(StrtVarA + 1, parse(0, EachParse), Chr$(93)): EndVarA = InStr(StrtVarA, parse(0, EachParse), Chr$(91))
  StrtVarB = InStr(StrtVarB + 1, parse(1, EachParse), Chr$(93)): EndVarB = InStr(StrtVarB, parse(1, EachParse), Chr$(91))
 x$ = Mid$(parse(0, EachParse), StrtVarA + 1, EndVarA - StrtVarA - 1)
 Y$ = Mid$(parse(1, EachParse), StrtVarB + 1, EndVarB - StrtVarB - 1)
```

```
If Len(x$) > 1 And Len(Y$) > 1 Then
 Lngth = Len(x$)
 Do 'LENGTH OF MOVING WINDOW
 ReDim MatchPos(-1 To 10, 4): ReDim MatchSet(10, 50, 1): StrtBit = 1
  Do 'MOVING THE WINDOW
   Bit$ = Mid$(x$, StrtBit, Lngth) 'WINDOW
   If InStr(1, Y$, Bit$) <> 0 Then 'IF WINDOW EXISTS IN Y$ THEN FIND IT
    FindMatch Y$, Bit$, StrtBit, FoundMatch, ReplyPos, MatchPos(), k
   End If
   StrtBit = StrtBit + 1
 Loop Until StrtBit + Lngth >= Len(x$) + 2 'END LOOP MOVING WINDOW
      If FoundMatch = True Then
'SHOWARRAY MatchPos(), k
       If k = 1 Then 'IF ONLY ONE MATCH OCCURS FOR A GIVEN LENGTH
        Replace parse(0, EachParse), StrtVarA + MatchPos(0, 0), Len(Bit$), parse(0, EachParse), parse(1, EachParse)
        Replace parse(1, EachParse), StrtVarB + MatchPos(0, 1), Len(Bit$), parse(1, EachParse), parse(0, EachParse)
        LegitMatch = True
       Else 'IF MORE THAN ONE MATCH OCCURS FOR A GIVEN LENGTH
        BufferA$ = parse(0, EachParse): BufferB$ = parse(1, EachParse)
        For Seq = 0 To k - 1 'FOR EACH START MATCH
         If MatchPos(Seq, 3) <> True Then
          Do 'LOOP FOR EACH SEQUENCE FOR EACH START MATCH
          SeqFound = False
          For NextSeq = Seq + 1 To k - 1 'FOR EACH SEQUENCE OF EACH START MATCH
            If MatchPos(NextSeq, 1) > MatchPos(Seq, 1) + Len(Bit$) - 1 Then 'IF (NEXT IS LARGER THAN END OF FIRST FOR $2
```

```
            If MatchPos(NextSeq, 0) > MatchPos(Seq, 0) + Len(Bit$) - 1 Then 'NEXT IS LARGER THAN END OF FIRST FOR $1
'LARGER THAN END OF LARGEST to DATE FOR $2
              If MatchPos(NextSeq, 1) > LastNum + Len(Bit$) - 1 And MatchPos(NextSeq, 0) > LastNum1 + Len(Bit$) - 1 Then
               If MatchPos(NextSeq, 2) <> True And MatchPos(LastNumRec, 4) <> True Then 'NOT VISITED IN $2
                MatchSet(0, r, 0) = MatchPos(Seq, 0): MatchSet(0, r, 1) = MatchPos(Seq, 1)
                MatchSet(q + 1, r, 0) = MatchPos(NextSeq, 0): MatchSet(q + 1, r, 1) = MatchPos(NextSeq, 1)
                MatchPos(NextSeq, 2) = True: MatchPos(NextSeq, 3) = True: MatchPos(Seq, 3) = True
                q = q + 1: SeqFound = True: LastNumRec = LastNum1: LastNum1 = MatchPos(NextSeq, 0)
                LastNum = MatchPos(NextSeq, 1): LegitMatch = True
               End If
                End If
               End If
              End If
'IF SEQUENCE IN $2 IS LEGIT BUT DOES NOT FORM PART OF A SEQUENCE BECAUSE OF CONSTRAINTSIN $1 THEN SET FLAG AND DELETE VISITED THIS
'SEQUENCE IN SUBSEQUENT ELEMENTS OF $2
              If MatchPos(NextSeq, 0) > LastNumRec + Len(Bit$) - 1 = False And MatchPos(NextSeq, 4) = True Then
               For ClearArray = NextSeq + 1 To k - 1
                MatchPos(ClearArray, 2) = False
               Next ClearArray
              End If
              If MatchPos(NextSeq, 0) > LastNumRec + Len(Bit$) - 1 = False Then
               MatchPos(LastNumRec, 4) = True: MatchPos(NextSeq, 4) = True '$1 constraint on an otherwise legit sequence
             End If
             LastNumRec = 0
            Next NextSeq 'LOOP NEXT MATCH AS CANDIDATE FOR SEQUENCE
```

```
      If SeqFound = True Then r = r + 1: q = 0: LastNum = 0: LastNum1 = 0
    Loop Until SeqFound = False 'END LOOP FOR THIS SEQUENCE OF GIVEN START MATCH
     For ClearArray = 0 To k - 1 'CLEAR VISITED FOR THIS SEQUENCE FLAGS
      MatchPos(ClearArray, 2) = False: MatchPos(ClearArray, 4) = False
     Next ClearArray
   End If
  Next Seq 'LOOP NEXT SEQUENCE START
End If
For ClearArray = 0 To k - 1 'RECORD NON-SEQUENTIAL MATCHES
 If Len(MatchPos(ClearArray, 3)) = 0 Then
  MatchSet(0, r, 0) = MatchPos(ClearArray, 0)
   MatchSet(0, r, 1) = MatchPos(ClearArray, 1)
   r = r + 1: LegitMatch = True
  End If
  MatchPos(ClearArray, 3) = False 'CLEAR NEVER BEEN PART OF A SEQUENCE
 Next ClearArray
'SHOWARRAY MatchSet(), r
 If k > 1 Then 'REPLACE AMBIGUOUS MATCHES
'REPLACE FIRST AMBIGUITY IN CURRENT PARSE STRING
  Do
   Replace parse(0, EachParse), StrtVarA + MatchSet(q, 0, 0) + Brkts, Len(Bit$), parse(0, EachParse), parse(1, EachParse)
   Replace parse(1, EachParse), StrtVarB + MatchSet(q, 0, 1) + Brkts, Len(Bit$), parse(1, EachParse), parse(0, EachParse)
   q = q + 1: Brkts = Brkts + 2
  Loop Until Len(MatchSet(q, 0, 0)) = 0
  Brkts = 0: q = 0
```

```
'FIND END OF PARSE LIST
        Do
         If Len(parse(0, EndList)) <> 0 Then EndList = EndList + 1
        Loop Until Len(parse(0, EndList)) = 0
'REPLACE SUBSEQUENT AMBIGUITIES AT END OF PARSE LIST
        For EachMatchSet = EndList To EndList + (r - 2)
         parse(0, EachMatchSet) = BufferA$: parse(1, EachMatchSet) = BufferB$
          Do
           Replace parse(0, EachMatchSet), StrtVarA + MatchSet(q, EachMatchSet - EndList + 1, 0) + Brkts, Len(Bit$), parse(0, EachMatchSet), parse(1, EachMatchSet)
           Replace parse(1, EachMatchSet), StrtVarB + MatchSet(q, EachMatchSet - EndList + 1, 1) + Brkts, Len(Bit$), parse(1, EachMatchSet), parse(0, EachMatchSet)
            q = q + 1: Brkts = Brkts + 2
           Loop Until Len(MatchSet(q, EachMatchSet, 1)) = 0
           Brkts = 0: q = 0
         Next EachMatchSet
        End If
        Brkts = 0: Erase MatchPos: Erase MatchSet: k = 0: Lngth = Lngth - 1: r = 0: EndList = 0: Exit Do
       End If
       Lngth = Lngth - 1
      Loop Until Lngth < 2 'LOOP MOVING WINDOWS TILL LENGTH < 2
      If LegitMatch = True Then StrtVarA = 0: StrtVarB = 0
      FoundMatch = False: LegitMatch = False: StrtBit = 1
    End If
   Loop Until InStr(StrtVarA + 1, parse(0, EachParse), Chr$(93)) = 0 Or InStr(StrtVarB + 1, parse(1, EachParse), Chr$(93)) = 0 'LOOP FOR EACH VARIABLE
   EachParse = EachParse + 1: StrtVarA = 0: StrtVarB = 0
Loop 'LOOP FOR EACH PARSE
```

```
 For RB = 0 To EachParse - 1 'REMOVE START/END BRACKETS
  RemoveHatch parse(0, RB)
  RemoveHatch parse(1, RB)
 Next RB

 If Bracketed = True Then
  For RB = 0 To EachParse - 1
   RemoveEmptyConstants parse(0, RB)
   RemoveEmptyConstants parse(1, RB)
  Next RB
 End If

End Sub
```

11.1.2.5 Variable Extractor

```
Sub ExtractVariables ()
'sub routine detects variables which exist when signatures are subtracted from them and stores
'them in a list such that if the list is placed directly underneath their corresponding pattern
'list they appear in the same positions

 If lstdb(0).ListCount <> 0 Then
  ReDim C$(-1 To 0) 'contains constants from patterns so they can be detected in utternaces
  ReDim VarStim$(3, -1 To 0) 'contains variables detected from utterance stimuli
  ReDim VarResp$(3, -1 To 0) 'contains variables detected from utterance responses
  ReDim DeleteUtterance(lstdb(0).ListCount - 1) 'utterance list number flag, true if utterance
                                'forms both a constant correspondence and an
                                'hypthosesisabe feature couplet with a pattern.
                                'Flag is used to delete utterances at end.

  'for each signature base
  For EachBase = 23 To 29
   TargetBase = -1
'for each signature pair in the current base
  Do Until EachPattern >= lstdb(EachBase).ListCount - 1
   NoCon1 = False: NoCon2 = False: NewPat = True: FirstAdd = True: NewPatternFormed = False
   ConCorrFound = False 'true if a set of textual couplets has produced and constant correspondence
'get stimulus and response pattern
```

```
        PatStim$ = lstdb(EachBase).List(EachPattern)
        PatResp$ = lstdb(EachBase).List(EachPattern + 1)
        If PatStim$ <> "" And PatResp$ <> "" Then
'transfer signature couplet to feature base
          lstdb(EachBase - 22).AddItem PatStim$
          lstdb(EachBase - 22).AddItem PatResp$
          lstdb(EachBase - 22).AddItem ""
'for each textual couplet
        For EachUtterance = 0 To lstdb(0).ListCount - 1 Step 3
          UttStim$ = lstdb(0).List(EachUtterance)

'detect whether constant equivalence exists between PatStim$ and UttStim$
            ConstantEquivalence PatStim$, UttStim$, ConCorr, NoCon1

'if utterance constant correspondence is true
          If ConCorr = True Then
           UttResp$ = lstdb(0).List(EachUtterance + 1)

'detect whether constant correspondence exists between PatStim$ and UttStim$
            ConstantEquivalence PatResp$, UttResp$, ConCorr, NoCon2
        End If

'if constant correspondence exists between stimulus + response patterns & utterances
        If ConCorr = True Then
          ConCorrFound = True
```

```
'mine for variables in stimulus signature
        VarMine UttStim$, PatStim$, VarStim$(), VarResp$(), NumVar, 0, LastNumVar
        LastNumVar = NumVar: NumVar = 0
'mine for variables in corresponding response signature
        VarMine UttResp$, PatResp$, VarResp$(), VarStim$(), NumVar, 1, LastNumVar
        DeleteUtterance(EachUtterance) = True
'if new pattern, then form buffer of stimulus and reponse pattern
        If NewPat Then
          StimTemp$ = lstdb(EachBase).List(EachPattern)
          RespTemp$ = lstdb(EachBase).List(EachPattern + 1)
          NewPat = False
        End If
        StimBuf$ = StimTemp$: RespBuf$ = RespTemp$

'construct stimulus feature in buffer
        Do
          StrtVar = InStr(StrtVar + 1, StimBuf$, "]")
          EndVar = InStr(StrtVar, StimBuf$, "[")
          StimBuf$ = Left$(StimBuf$, StrtVar - 1) + VarStim$(1, EachVarSlot) + Right$(StimBuf$, Len(StimBuf$) - EndVar)
          StrtVar = StrtVar + Len(VarStim$(1, EachVarSlot)): EachVarSlot = EachVarSlot + 1
        Loop Until InStr(StrtVar, StimBuf$, "]") = 0
        StrtVar = 0: EndVar = 0

'construct corresponding response featuer in buffer
        Do
```

```
    StrtVar = InStr(StrtVar + 1, RespBuf$, "]")
    EndVar = InStr(StrtVar, RespBuf$, "[")
    RespBuf$ = Left$(RespBuf$, StrtVar - 1) + VarStim$(1, EachVarSlot) + Right$(RespBuf$, Len(RespBuf$) - EndVar)
    StrtVar = StrtVar + Len(VarStim$(1, EachVarSlot)): EachVarSlot = EachVarSlot + 1
  Loop Until InStr(StrtVar, RespBuf$, "]") = 0
  EachVarSlot = 0: StrtVar = 0: EndVar = 0

'check whether pattern already exists in current feature base
  ExistPat = False
  For EachPat = 0 To lstdb(EachBase - 22).ListCount - 1 Step 3
'if constructed stimulus and response pattern are same as already in currentfeature base then set ExistsPat to true and increment EachPat: exit loop
    If StimBuf$ = lstdb(EachBase - 22).List(EachPat) And RespBuf$ = lstdb(EachBase - 22).List(EachPat + 1) Then
      ExistPat = True: EachPat = EachPat + 3: Exit For
    End If
  Next EachPat
  EachPat = EachPat - 3
'if constructed feature couplet exist in feature base
  If ExistPat Then
'construct variable cluster and check whether it already exists in variable base
    SBuf$ = SBuf$ & "/"
    For EachVar = 0 To LastNumVar
      SBuf$ = SBuf$ & VarStim$(2, EachVar)
    Next EachVar
    RBuf$ = RBuf$ & "/"
    For EachVar = LastNumVar + 1 To LastNumVar + NumVar + 1
```

```
            RBuf$ = RBuf$ & VarStim$(2, EachVar)
        Next EachVar
'if stimulus and response do not contain variable cluster to be added
        If InStr(1, lstdb(EachBase - 14).List(EachPat), SBuf$) And InStr(1, lstdb(EachBase - 14).List(EachPat + 1), RBuf$) Then
        Else
'add variables to appropriate variable list
            lstdb(EachBase - 14).List(EachPat) = lstdb(EachBase - 14).List(EachPat) + SBuf$

            lstdb(EachBase - 14).List(EachPat) = lstdb(EachBase - 14).List(EachPat) + "~"

            lstdb(EachBase - 14).List(EachPat + 1) = lstdb(EachBase - 14).List(EachPat + 1) + RBuf$

            lstdb(EachBase - 14).List(EachPat + 1) = lstdb(EachBase - 14).List(EachPat + 1) + "~"
        End If
        SBuf$ = "": RBuf$ = ""
'if constructed stimulus and response patterns are not already in current signature base
        Else
'if this is the first time that a signature is being added for this set of textual
'coupets then replace the existing feature couplet in current feature baseand add variables
        If FirstAdd = True Then
         NewPatternFormed = True
'add feature couplet to current feature base
        lstdb(EachBase - 22).List(EachPat) = StimBuf$
        lstdb(EachBase - 22).List(EachPat + 1) = RespBuf$
'if this signature comes from an earlier pattern then make room on feature base
```

```
            If lstdb(EachBase).List(EachPat) <> PatStim$ Then
'add space at end of feature base list
             For X = 0 To 2
              lstdb(EachBase).AddItem ""
             Next X
'move signature base list down three lines
             For X = lstdb(EachBase).ListCount - 1 To EachPat + 4 Step -1
              lstdb(EachBase).List(X - 1) = lstdb(EachBase).List(X - 4)
              lstdb(EachBase).List(X - 4) = ""
             Next X
            End If
             If lstdb(EachBase - 14).ListCount - 1 < EachPat + 2 Then
              Do Until lstdb(EachBase - 14).ListCount - 1 = EachPat + 2
               lstdb(EachBase - 14).AddItem ""
              Loop
             End If
'add varibles
            lstdb(EachBase - 14).List(EachPat) = lstdb(EachBase - 14).List(EachPat) + "/"
            For EachVar = 0 To LastNumVar
             lstdb(EachBase - 14).List(EachPat) = lstdb(EachBase - 14).List(EachPat) + VarStim$(2, EachVar)
            Next EachVar
            lstdb(EachBase - 14).List(EachPat) = lstdb(EachBase - 14).List(EachPat) + "~"

            lstdb(EachBase - 14).List(EachPat + 1) = lstdb(EachBase - 14).List(EachPat + 1) + "/"
            For EachVar = LastNumVar + 1 To LastNumVar + NumVar + 1
```

```
            lstdb(EachBase - 14).List(EachPat + 1) = lstdb(EachBase - 14).List(EachPat + 1) + VarStim$(2, EachVar)
          Next EachVar
          lstdb(EachBase - 14).List(EachPat + 1) = lstdb(EachBase - 14).List(EachPat + 1) + "~"
          FirstAdd = False
'if this is not the first time that a feature couplet is being added for this set of textual coupets then add the Feature couplet to the end of current Featurecouplet base and add
'variables
        Else
          NewPatternFormed = True
'add feature couplet to current feature base
          lstdb(EachBase - 22).AddItem StimBuf$
          lstdb(EachBase - 22).AddItem RespBuf$
          lstdb(EachBase - 22).AddItem ""
'add room in current variable base to accept variables
          lstdb(EachBase - 14).AddItem ""
          lstdb(EachBase - 14).AddItem ""
          lstdb(EachBase - 14).AddItem ""
'add space in signature base for multi formation of features
          lstdb(EachBase).AddItem ""
          lstdb(EachBase).AddItem ""
          lstdb(EachBase).AddItem ""
'move feature base list down three lines
        For X = lstdb(EachBase).ListCount - 1 To EachPat + 6 Step -1
          lstdb(EachBase).List(X - 1) = lstdb(EachBase).List(X - 4)
          lstdb(EachBase).List(X - 4) = ""
        Next X
```

```
'add varibles
            lstdb(EachBase - 14).List(EachAddition) = lstdb(EachBase - 14).List(EachAddition) + "/"
            For EachVar = 0 To LastNumVar
              lstdb(EachBase - 14).List(EachAddition) = lstdb(EachBase - 14).List(EachAddition) + VarStim$(2, EachVar)
            Next EachVar
             lstdb(EachBase - 14).List(EachAddition) = lstdb(EachBase - 14).List(EachAddition) + "~"

             lstdb(EachBase - 14).List(EachAddition + 1) = lstdb(EachBase - 14).List(EachAddition + 1) + "/"
            For EachVar = LastNumVar + 1 To LastNumVar + NumVar + 1
             lstdb(EachBase - 14).List(EachAddition + 1) = lstdb(EachBase - 14).List(EachAddition + 1) + VarStim$(2, EachVar)
            Next EachVar
              lstdb(EachBase - 14).List(EachAddition + 1) = lstdb(EachBase - 14).List(EachAddition + 1) + "~"
            End If
            If NewPatternFormed = True Then
             EachAddition = EachAddition + 3
             NewPatternFormed = False
            End If
           End If
           ReDim VarResp$(3, -1 To 0): ReDim VarStim$(3, -1 To 0): NumVar = 0
         End If
       Next EachUtterance
       'else if there was no constant correspondence with any utterance or the variables generated went to another signature then delete candidate feature
       'from currentsignature base
       If lstdb(EachBase - 14).List(lstdb(EachBase - 22).ListCount - 2) = "" Then
         lstdb(EachBase - 22).RemoveItem lstdb(EachBase - 22).ListCount - 1
```

```
      lstdb(EachBase - 22).RemoveItem lstdb(EachBase - 22).ListCount - 1
      lstdb(EachBase - 22).RemoveItem lstdb(EachBase - 22).ListCount - 1
    End If
   End If 'If PatStim$ <> "" And PatResp$ <> ""
   EachPattern = EachPattern + 3
  Loop 'each pattern
  EachPattern = 0: EachAddition = 0
Next EachBase

If DeleteCouplets = True Then
 'delete utterances used to generate variables
   For deleteable = lstdb(0).ListCount - 3 To 0 Step -3
    If DeleteUtterance(deleteable) = True Then
     lstdb(0).RemoveItem deleteable
     lstdb(0).RemoveItem deleteable
     lstdb(0).RemoveItem deleteable
    End If
   Next deleteable
  End If
End If

End Sub
```

11.1.3 Pattern Comparitor:

11.1.3.1 ConstantEquivalence Detector (NonfuzzBabe).

```
Sub SignatureEquivalence (ProtoFeatures$(), SizeOfProtoFeatures, FoundConEquiv)


UttSig$ = "]" + Utterance$ + "["
FoundConEquiv = False

 'for each signature base
  For EachBase = 1 To 7
 'for each signature in each base
    For EachPattern = 0 To lstdb(EachBase).ListCount - 1
'get each pattern
     Pat$ = lstdb(EachBase).List(EachPattern)
     If Pat$ <> "" Then
'test for signature equivalence with input utterance
      ConstantEquivalence Pat$, Utterance$, ConEquiv, NoCon
'if signature equivalence exists then add location of sig to list 8 and generate protovars
      If ConEquiv = True Then
       FoundConEquiv = True
'load location into list8
       list8(0).AddItem EachBase
```

```
        list8(1).AddItem EachPattern
        list8(2).AddItem "1.0"
'load location and protofeature into ProtoFeatures$()
        ReDim Preserve ProtoFeatures$(2, EachEquivalence)
        ProtoFeatures$(0, EachEquivalence) = EachBase
        ProtoFeatures$(1, EachEquivalence) = EachPattern
'manufacture protovar in Var$
        Var$ = "/"
        StrtCon = 0: EndCon = 0: StrtCon1 = 0
        Do Until InStr(StrtCon + 1, Pat$, "[") = 0
          StrtCon = InStr(StrtCon + 1, Pat$, "[")
          EndCon = InStr(StrtCon + 1, Pat$, "]")
'test whether a complete constant exists that is not a nul
          If StrtCon <> 0 And EndCon <> 0 And EndCon - StrtCon <> 1 Then
            Con$ = Mid$(Pat$, StrtCon + 1, EndCon - StrtCon - 1)
            StrtCon1 = InStr(StrtCon1 + 1, UttSig$, Con$)
            UttSig$ = Left(UttSig$, StrtCon1 - 1) + "[" + Con$ + "]" + Right$(UttSig$, Len(UttSig$) - (StrtCon1 + Len(Con$)) + 1)
            StrtCon1 = StrtCon1 + Len(Con$)
          End If
        Loop
        ProtoFeatures$(2, EachEquivalence) = UttSig$
        EachEquivalence = EachEquivalence + 1
        UttSig$ = "]" + Utterance$ + "["
      End If
    End If
```

```
    Next EachPattern
  Next EachBase
  SizeOfProtoFeatures = EachEquivalence - 1

End Sub
```

11.1.3.2 Correspondence Detector (CorrBabe)

```
Sub SignatureCorrespondence (ProtoFeatures$(), SizeOfProtoFeatures, FoundConCorr)

 ReDim MaxConstantLngth(0):  ReDim Target$(0): ReDim ListNo(0): ReDim BaseNo(0)

 FoundConCorr = False
 MaxConstantLngth(0) = -1

'for each signature base
  For EachBase = 1 To 7
'for each signature in each base
    For EachPattern = 0 To lstdb(EachBase).ListCount - 1
'get each pattern
      Pat$ = lstdb(EachBase).List(EachPattern)
      If Pat$ <> "" Then
'calculate the degree of correspondence between current signature and input utterance
       CalcConstantCorrespondence Utterance$, Pat$, ConCorr, UttSig$
       ThisConLngth = ConCorr
'if constant correspondence equals one already found then add info to buffer arrays
      If ThisConLngth = MaxConstantLngth(0) Then
       SameLngth = SameLngth + 1
      ReDim Preserve MaxConstantLngth(SameLngth): MaxConstantLngth(SameLngth) = ThisConLngth
      ReDim Preserve ListNo(SameLngth): ListNo(SameLngth) = EachPattern
```

```
        ReDim Preserve BaseNo(SameLngth): BaseNo(SameLngth) = EachBase
        ReDim Preserve Target$(SameLngth): Target$(SameLngth) = UttSig$
      End If
'if constant correspondence is greater than largest already found then create new buffer arrays
    If ThisConLngth > MaxConstantLngth(0) And ThisConLngth <> 0 Then
      ReDim MaxConstantLngth(0): ReDim ListNo(0): ReDim BaseNo(0): SameLngth = 0
      MaxConstantLngth(0) = ThisConLngth
      Target$(0) = UttSig$: ListNo(0) = EachPattern: BaseNo(0) = EachBase
      FoundConCorr = True
    End If
  End If
    ThisConLngth = 0
  Next EachPattern
 Next EachBase

 ReDim ProtoFeatures$(2, SameLngth)
'transfer contents of buffer arrays for winning correspondence into list8 and protofeatures$()
 For EachMaxCorr = 0 To SameLngth
  list8(0).AddItem BaseNo(EachMaxCorr)
  list8(1).AddItem ListNo(EachMaxCorr)
  list8(2).AddItem "1.0"
  ProtoFeatures$(0, EachMaxCorr) = BaseNo(EachMaxCorr)
  ProtoFeatures$(1, EachMaxCorr) = ListNo(EachMaxCorr)
  ProtoFeatures$(2, EachMaxCorr) = Target$(EachMaxCorr)
Next EachMaxCorr
```

```
SizeOfProtoFeatures = SameLngth

End Sub
```

11.1.3.3 FuzzyCorrespondence Detector (FuzzBabe)

```
Sub EvaluateConstantCorrespondence (ProtoFeatures(), SizeOfFuzzyConCorr, FoundConCorr)
'this procedure compares the input utterance with each signature in each base and measures
'the degree of constant correspondence in two ways as follows:

'Data accumulated in FuzzySignatureConCorr()
        '(0, 0-n) - location of each winning correspondence (base)
        '(1, 0-n) - location of each winning correspondence (list)
        '(2, 0-n) - the ratio of correspondence to individual pattern lengths

'Data accumulated in FuzzyGlobalConCorr()
        '(0, 0-n) - location of each winning correspondence (base)
        '(1, 0-n) - location of each winning correspondence (list)
        '(2, 0-n) - 'the absolute constant correspondence between input
                'utterance and all signatures which share correspondence. This
                'data is latter normalised to a fuzzy set of interval[0-1]

'if a constant correspondence exists then a ProtoFeature is generated and stored in
'ProtoFeatures() as follows:
        '(0, 0-n) - location of signature from which hypothetic signature was formed (base)
        '(1, 0-n) - location of signature from which hypothetic signature was formed (list)
        '(2, 0-n) - the prototypical signature string
correspondence = False
```

```
 FoundConCorr = False
'for each signature base
 For EachBase = 1 To 7
'for each signature of current base
   For EachSig = 0 To lstdb(EachBase).ListCount - 1
'if the list position is not a blank line (done for cosmetic purposes to make reading easier)
     If lstdb(EachBase).List(EachSig) <> "" Then
'get each signature of each base
       signature$ = lstdb(EachBase).List(EachSig)
'evaluate the correspondence ratio between input utterance and current signature
       CalcSignatureConstantCorrespondence signature$, SignatureConCorr, AbsoluteCorr, UttSig$

 If SignatureConCorr <> 0 Then
        ReDim Preserve FuzzySignatureConCorr(2, EachCorr)
        ReDim Preserve FuzzyGlobalConCorr(2, EachCorr)
        ReDim Preserve ProtoFeatures(2, EachCorr)
        ReDim Preserve GlobalConCorr(EachCorr)
'store the resulting protofeature and its derived location
        ProtoFeatures(0, EachCorr) = EachBase
        ProtoFeatures(1, EachCorr) = EachSig
        ProtoFeatures(2, EachCorr) = UttSig$
'store local correspondence and its derived location
       FuzzySignatureConCorr(0, EachCorr) = EachBase
       FuzzySignatureConCorr(1, EachCorr) = EachSig
      FuzzySignatureConCorr(2, EachCorr) = SignatureConCorr
```

```
'store global correspondence and its derived location
        FuzzyGlobalConCorr(0, EachCorr) = EachBase
        FuzzyGlobalConCorr(1, EachCorr) = EachSig
        GlobalConCorr(EachCorr) = AbsoluteCorr
        EachCorr = EachCorr + 1
        FoundConCorr = True
      End If
    End If
  Next EachSig
Next EachBase

'if any constant correspondences where found then record the quantity and normalize
'global correspondence to a fuzzy set in the interval [0-1]
 If FoundConCorr = False Then
   SizeOfFuzzyConCorr = 0
 Else
   SizeOfFuzzyConCorr = EachCorr - 1
   Normalize GlobalConCorr(), SizeOfFuzzyConCorr
   For EachElement = 0 To EachCorr - 1
     FuzzyGlobalConCorr(2, EachElement) = GlobalConCorr(EachElement)
   Next EachElement
 End If

End Sub
```

11.1.3.4 SpreadingActivation Generator

```
Sub GenerateActivatedResponseFeatures (ProtoFeatures(), SizeOfProtoFeatures)
'the purpose of this procedure is to:
'detect and store couplet daughters of signatures activated directly by the input utterance
'detect and store signatures which are identical to these daughters
'allocate the Protofeatures of the directly activated daughters to the identical daughters

'The list of response locations is stored in list8 as follows:
        'list12(0)- base location
        'list12(1)- list location
        'list12(2)- likelihood donated by stimulus activated signature
'the list of Protofeatures is stored in ProtoFeatures() as follows:
        'ProtoFeatures(0, n) = List number
        'ProtoFeatures(1, n) = Location in list
        'ProtoFeatures(2, n) = Allocated protofeature string

'transfer location and constant correspondence information of the couplet daughters of
'signatures activated directly by the input stimulus. This action is taken first so that
'protofeatures order matches list 13 order
'also load contents of list12 into NewlyActivated()
ReDim Preserve NewlyActivated(2, SizeOfProtoFeatures)
For EachElement = 0 To list8(0).ListCount - 1
 list12(0).AddItem list8(0).List(EachElement)
 If Val(list8(1).List(EachElement)) / 3 = Int(Val(list8(1).List(EachElement)) / 3) Then
```

```
    list12(1).AddItem list8(1).List(EachElement) + 1
   Else
    list12(1).AddItem list8(1).List(EachElement) - 1
   End If
   list12(2).AddItem Format$(list8(2).List(EachElement), ".0####")
   NewlyActivated(0, EachElement) = Format$(list8(0).List(EachElement))
   NewlyActivated(1, EachElement) = Format$((list8(1).List(EachElement)))
   NewlyActivated(2, EachElement) = Format$(list8(2).List(EachElement))
Next EachElement

'repeat for each newly activated pattern
 Do
'get the couplet daugther of each signature activated directly by the input stimulus
   For EachResp = 0 To NumActivated
    If NewlyActivated(1, EachResp) / 3 = Int(NewlyActivated(1, EachResp) / 3) Then
     RespSig$ = lstdb(NewlyActivated(0, EachResp)).List(NewlyActivated(1, EachResp) + 1)
     StimElement = True
    Else
     RespSig$ = lstdb(NewlyActivated(0, EachResp)).List(NewlyActivated(1, EachResp) - 1)
     StimElement = False
   End If
   NumActivated = 0
'convert couplet daughter to a feature
   StripVarNos RespSig$, RespSig$
   If RespSig$ <> "" Then
```

```
'store location of any features which are identical to current activated response feature
      For EachBase = 1 To 7
       For EachSig = 0 To lstdb(EachBase).ListCount - 1
        ActivCand$ = lstdb(EachBase).List(EachSig)
        StripVarNos ActivCand$, ActivCand$
'if response feature = candidate feature
        If RespSig$ = ActivCand$ Then
'check whether location already exists in activated response list
         Duplicate = False
         For dup = 0 To list12(0).ListCount - 1
          If Val(list12(0).List(dup)) = EachBase Then
           If Val(list12(1).List(dup)) = EachSig Then
            Duplicate = True: Exit For
           End If
          End If
         Next dup
'if daughter location does not already exist in the activated response list then add it to the activated response list
         If Duplicate = False Then
          list12(0).AddItem EachBase
          list12(1).AddItem EachSig
          list12(2).AddItem Format$(NewlyActivated(2, EachResp), ".0####")
'add newly activated location to NewlyActivated() so that its daughter feature can be searched for to continue the spreading activation process
          ReDim Preserve BuffNewlyActivated(2, NumActivated)
          BuffNewlyActivated(0, NumActivated) = EachBase
          BuffNewlyActivated(1, NumActivated) = EachSig
```

```
            BuffNewlyActivated(2, NumActivated) = NewlyActivated(2, EachResp)
            NumActivated = NumActivated + 1
'allocate the activated response signature the protofeature of the signature directly activated by the input utterance whose daughter couplet is identical with the current activated
'response signature to the ProtoFeature array
            SizeOfProtoFeatures = SizeOfProtoFeatures + 1
            ReDim Preserve ProtoFeatures(2, SizeOfProtoFeatures)
            ProtoFeatures(0, SizeOfProtoFeatures) = EachBase
            ProtoFeatures(1, SizeOfProtoFeatures) = EachSig
            ProtoFeatures(2, SizeOfProtoFeatures) = ProtoFeatures(2, EachResp)
          Else
            Duplicate = False
            End If
          End If
        Next EachSig
      Next EachBase
    End If
  Next EachResp

'transfer BuffNewlyActived() into NewlyActivated()
  ReDim NewlyActivated(2, NumActivated)
  For EachActivation = 0 To NumActivated - 1
    NewlyActivated(0, EachActivation) = BuffNewlyActivated(0, EachActivation)
    NewlyActivated(1, EachActivation) = BuffNewlyActivated(1, EachActivation)
    NewlyActivated(2, EachActivation) = BuffNewlyActivated(2, EachActivation)
  Next EachActivation
```

```
  Loop Until NumActivated = 0

End Sub
```

11.1.4 Response Hypothesiser:

11.1.4.1 CompetitiveResponse Constructor & ConversantInteraction Manager

```
Sub RESPOND (ProtoFeatures(), ActivatedClusters(), SizeOfActivatedClusters)

'construct prototypical reply based upon patterns activated by constant correspondence
 For EachOutput = 0 To list12(0).ListCount - 1
   HotBase = list12(0).List(EachOutput) 'stores base in which hotftrcplt exists
   HotFtrCplt = list12(1).List(EachOutput)'store list number of feature couplet with constant correspondence
 'form reply using input utterance and defuzzified pattern selection
    FormReplyCon Reply$, EachOutput
    lstoutput.AddItem Reply$
  Next EachOutput

'construct prototypical reply based upon patterns activated by variable correspondence
  For EachOutput = 0 To list3(0).ListCount - 1
    HotBase = list3(0).List(EachOutput) 'stores base in which hotftrcplt exists
    HotFtrCplt = list3(1).List(EachOutput)'store list number of first feature couplet with constant correspondence
'form reply using direct variable substitution in pattern fired by varaible correspondence
   For EachReply = 0 To SizeOfActivatedClusters
'if HotFtrCplt is a stimulus then return pattern overwritten with variable cluster plus  acceptable response in the past
     If HotFtrCplt / 3 = Int(HotFtrCplt / 3) Then
       FormReplyVar ActivatedClusters(), Reply$, EachReply
```

```
    lstoutput.AddItem Reply$
    FormOldResp ActivatedClusters(), Reply$, EachReply
    lstoutput.AddItem Reply$
   Else 'if HotFtrCplt is a response then return only pattern overwritten with variable cluster
    FormReplyVar ActivatedClusters(), Reply$, EachReply
    lstoutput.AddItem Reply$
   End If
  Next EachReply
Next EachOutput

'if output list does not contain "How should I reply? then add it
 HSIR = False
 For EachReply = 0 To lstoutput.ListCount - 1
  If lstoutput.List(EachReply) = "How should I reply?" Then
   HSIR = True: HSIRNo = EachReply: Exit For
  End If
 Next EachReply
 If HSIR = False Then
  lstoutput.AddItem "How should I reply?"
  HSIRNo = EachReply:
 Else
  HSIR = False
End If

'expand lstoutput to fit text
```

```
 If lstoutput.ListCount > 2 Then
  If lstoutput.ListCount * 270 < 1500 Then
   lstoutput.Height = (lstoutput.ListCount * 270)
  Else
   lstoutput.Height = 1500
  End If
 End If
'make ordered list and number list visible
 lstNums.Visible = True
 lstOrderedResponses.Visible = True
 label42.Visible = True
 label43.Visible = True
 label44.Visible = True
'clear selected in lstoutput
 For X = 0 To lstoutput.ListCount - 1
  lstoutput.Selected(X) = False
 Next X
 GenerateWinningResponse OutputlistNum, CandidateFound
 If CandidateFound = True Then
  txtOutput.Text = lstoutput.List(OutputlistNum): SelectText
 Else
  txtOutput.Text = "How should I reply?": SelectText
  OutputlistNum = HSIRNo
 End If
 picture1.Visible = True 'non acceptable picture box
```

```
 picture1.Print "None appropriate?"

'if menu item "Accept System Responses" is checked then set focus to txtoutput
 If UserSelect = False Then txtOutput.SetFocus
 If NonAcceptable = True Then picture1.SetFocus

'bodge to solve VB fault where eventhough nothing is seleced in a list box, listindex  returns an index one larger than the listcount?????
 Do
   UserSelectReply Auto
 Loop Until lstoutput.List(lstoutput.ListIndex) <> "" Or Auto = "SystemWinner" Or Auto = "SytemCandidate" Or Auto = "NoWin"
   picture1.Visible = False
   picture1.Cls
   lstNums.Visible = False
   lstOrderedResponses.Visible = False
   label42.Visible = False
   label43.Visible = False
   label44.Visible = False
 'if (by inference) system response was the added "How should I reply?"
  If OutputlistNum > list4(0).ListCount Then Auto = "NoWin"
 'if sytem response was appropriate
  If Auto = "SystemWinner" Then
   ThisReply$ = lstoutput.List(OutputlistNum)
   If ThisReply$ <> "How should I reply?" Then
    LastUtteranceWasAnHypothesis = True
'relate hotbase and hotfircplt with human selection
```

```
    HotBase = list4(0).List(OutputlistNum)
    HotFtrCplt = list4(1).List(OutputlistNum)
   Else
    LastUtteranceWasAnHypothesis = False
   End If
   Hypothesis$ = ThisReply$: STM$ = Utterance$
'clear selected in lstoutput
   lstoutput.Clear
   lstoutput.Height = 630
   UtterancesAdded = UtterancesAdded + 2
  lstdb(0).AddItem Utterance$: lstdb(0).AddItem ThisReply$: label26.Visible = False: SelectText
  ElseIf Auto = "NoWin" Then 'no system competitive hypothesis was appropriate
   ThisReply$ = "How should I reply?"
   LastUtteranceWasAnHypothesis = False
   Hypothesis$ = ThisReply$: STM$ = Utterance$
   HotBase = -1
   HotFtrCplt = -1
'clear selected in lstoutput
   lstoutput.Clear
   lstoutput.Height = 630
   txtOutput.Text = ThisReply$: SelectText
  UtterancesAdded = UtterancesAdded + 2
  lstdb(0).AddItem Utterance$: lstdb(0).AddItem ThisReply$: label26.Visible = False

 ElseIf Auto = "SystemCandidate" Then 'if system response was inappropriate use conversant selected response as appropriate reply
```

```
    ThisReply$ = lstoutput.List(lstoutput.ListIndex)
    txtOutput.Text = ThisReply$: SelectText
    If ThisReply$ = "How should I reply?" Then
      LastUtteranceWasAnHypothesis = False
    Else
      LastUtteranceWasAnHypothesis = True
    End If
    Hypothesis$ = ThisReply$: STM$ = Utterance$
'relate hotbase and hotftrcplt with human selection
    If ThisReply$ <> "How should I reply?" Then
      HotBase = list4(0).List(lstoutput.ListIndex)
      HotFtrCplt = list4(1).List(lstoutput.ListIndex)
    End If
'clear selected in lstoutput
    For X = 0 To lstoutput.ListCount - 1
      lstoutput.Selected(X) = False
    Next X
    lstoutput.Clear
    lstoutput.Height = 630
    UtterancesAdded = UtterancesAdded + 2
    lstdb(0).AddItem Utterance$: lstdb(0).AddItem ThisReply$: label26.Visible = False
  End If

End Sub
```

11.1.4.2 WinningResponse detector

```
Sub GenerateWinningResponse (OutputlistNum, CandidateFound)
'this procedure takes the list of competitive prototypical responses from lstoutput and computes
'the most frequently occuring full textual hypothesises as the winning response that is not
'"How should I reply?" or end in "?"

'**** at the moment, if there are more than one of such replies, procedure
'**** chooses the one nearest to the top of the list
 CandidateFound = False
'generate list of occurence of each prototypical response and load into frequencyData()
 SizeOfFrequencyData = lstoutput.ListCount - 1
 ReDim FrequencyData(1, SizeOfFrequencyData) 'column 0 represents frequency, column 1 represents already counted flag
 For EachHypo = 0 To SizeOfFrequencyData
  If FrequencyData(1, EachHypo) <> True Then 'not been counted before
   If lstoutput.List(EachHypo) <> "How should I reply?" Then 'not HSIR
    If InStr(1, lstoutput.List(EachHypo), "(...)") = 0 Then 'not (...)
     If lstoutput.List(EachHypo) <> txtInput.Text Then 'not same as input utterance
      cnt = cnt + 1
      CandidateFound = True
      FrequencyData(0, EachHypo) = cnt
      FrequencyData(1, EachHypo) = True
      For EachOtherHypo = EachHypo + 1 To SizeOfFrequencyData
       If lstoutput.List(EachHypo) = lstoutput.List(EachOtherHypo) Then
        If FrequencyData(1, EachOtherHypo) <> True Then
```

```
            cnt = cnt + 1
            FrequencyData(0, EachHypo) = cnt
            FrequencyData(1, EachOtherHypo) = True
          End If
        End If
      Next EachOtherHypo
    End If
  End If
 End If
End If
  cnt = 0
Next EachHypo


'select the winning competitor(s)
  'detect the set of most frequent responses into FrequentSet()
For EachComp = 0 To SizeOfFrequencyData
 If FrequencyData(0, EachComp) = MaxFreq And MaxFreq <> "" Then
  EachEl = EachEl + 1: ReDim Preserve FrequentSet(EachEl)
  FrequentSet(EachEl) = EachComp
End If
If FrequencyData(0, EachComp) > MaxFreq Then
 MaxFreq = FrequencyData(0, EachComp)
 ReDim FrequentSet(0): EachEl = 0
FrequentSet(EachEl) = EachComp
```

```
    replyfound = True
  End If
Next EachComp

'if any most frequency responses were detected then detect first most frequent response that does not end with "?" else return first mostfrequent response
If EachEl <> "" Then
 For EachResp = 0 To EachEl
  If InStr(1, lstoutput.List(FrequentSet(EachResp)), "?") = 0 Then
   OutputlistNum = FrequentSet(EachResp)
   Exit For
  End If
   Next EachResp
   If OutputlistNum = "" Then OutputlistNum = FrequentSet(0)
 End If
 'transfer ordered list of full replies to lstOrderedResponse
 'find min frequency
 MinFreq = MaxFreq
For EachComp = 0 To SizeOfFrequencyData
 If FrequencyData(0, EachComp) < MinFreq Then
  If FrequencyData(0, EachComp) <> "" Then
   MinFreq = FrequencyData(0, EachComp)
  End If
 End If
Next EachComp
```

```
MaxFreqBuff = MaxFreq
Do Until MaxFreqBuff = MinFreq - 1
 For EachComp = 0 To SizeOfFrequencyData
  If FrequencyData(0, EachComp) <> "" Then
   If FrequencyData(0, EachComp) = MaxFreqBuff Then
    lstOrderedResponses.AddItem lstoutput.List(EachComp)
    lstNums.AddItem MaxFreqBuff
    Change1 = True
   End If
  End If
 Next EachComp
 MaxFreqBuff = MaxFreqBuff - 1
Loop


If replyfound = True Then
 Exit Sub
Else
 'find topmost (...)
 For EachComp = 0 To SizeOfFrequencyData
  If InStr(1, lstoutput.List(EachComp), "(...)") <> 0 Then
   OutputlistNum = EachComp
   replyfound = True
  End If
```

```
    Next EachComp

  End If

  If replyfound = True Then
    Exit Sub
  Else
    OutputlistNum = 0
  End If

End Sub
```

11.2 Appendix B. (The MacWhinney corpus)

```
/*****************************************************************/
/*  Online appendix 1 for JAIR                                   */
/*  Original verb data file from MacWhinney                      */
/*  Note: one of multiple past tenses (hang-hung,-hanged) is removed */
/*  First column: spelling form;                                 */
/*  second: b for base, d for past tense, n for past participle,
*/
/*        z for third person singular, g for present participle  */
/*  third: 0 for regular, 1 for irregular;                      */
/*  fourth:  the Francis-Kucera frequency.
/*****************************************************************/
```

abandon b 0 15
abandoned d 0 5
abandoned n 0 20
abandoning g 0 7
abet b 0 0
abetted d 0 2
abetted n 0 2
abide b 0 7
abides z 0 2
abiding g 0 5
abolish b 0 8
abolished n 0 2
abound b 0 1
abounded d 0 2
absent b 0 2
absorb b 0 13
absorbed d 0 2
absorbed n 0 22
absorbing g 0 3
abstract b 0 3
abstracted n 0 2
abstracting g 0 3
abuse b 0 3
abused n 0 5
accelerate b 0 5
accelerated n 0 12
accelerating g 0 6
accept b 0 72
accepted d 0 28
accepted n 0 68
accepting g 0 19
accepts z 0 6
accommodate b 0 14
accommodates z 0 2
accommodating g 0 2
accompany b 0 8
accompanied d 0 8
accompanied n 0 29
accompanying g 0 17
accomplish b 0 24
accomplished d 0 4
accomplished n 0 39
accomplishing g 0 3
accord b 0 2
accorded n 0 4
according g 0 2
account b 0 27
accounted d 0 4
accounting g 0 8
accounts z 0 9
accrue b 0 0
accrued n 0 2
accruing g 0 7
accumulate b 0 3
accumulated n 0 10
accumulates z 0 2
accumulating g 0 3
accuse b 0 10
accused d 0 5
accused n 0 20
accuses z 0 2
accusing g 0 8
ache b 0 1
ached d 0 3
aching g 0 6
achieve b 0 51
achieved d 0 12
achieved n 0 50
achieves z 0 5
achieving g 0 15
acknowledge b 0 12
acknowledged d 0 3
acknowledged n 0 9
acknowledges z 0 2
acquaint b 0 3
acquainted n 0 12
acquiesce b 0 3
acquire b 0 27
acquired d 0 8
acquired n 0 18
acquires z 0 2
acquiring g 0 11
act b 0 74
acts z 0 11
acted d 0 11
acted n 0 7
acting g 0 56
activate b 0 2
activated n 0 5
adapt b 0 5
adapted n 0 12
adapting g 0 2
add b 0 88
added d 0 81
added n 0 91
adds z 0 10
adding g 0 21
address b 0 8
addressed d 0 7
addressed n 0 12
addresses z 0 4
addressing g 0 9
adhere b 0 4
adhered n 0 4
adjoin b 0 0
adjoined d 0 2
adjoining g 0 13
adjoins z 0 2
adjourn b 0 0
adjourned d 0 2
adjust b 0 16
adjusted d 0 3
adjusted n 0 30
adjusting g 0 11
adjusts z 0 2
administer b 0 3
administered n 0 13
administering g 0 4
admire b 0 10
admired d 0 9

admired	n 0 8	agreeing	g 0 7
admiring	g 0 4	agrees	z 0 11
admit	b 0 37	aid	b 0 22
admits	z 0 2	aided	d 0 2
admitted	d 0 25	aided	n 0 9
admitted	n 0 19	aiding	g 0 7
admitting	g 0 8	aids	z 0 6
admonish	b 0 0	aim	b 0 10
admonished	d 0 2	aimed	d 0 10
adopt	b 0 13	aimed	n 0 14
adopted	d 0 11	aiming	g 0 5
adopted	n 0 34	aims	z 0 3
adopting	g 0 11	alarm	b 0 2
adopts	z 0 2	alarmed	n 0 8
adore	b 0 2	alert	b 0 5
advance	b 0 15	alerted	n 0 2
advanced	d 0 9	alerting	g 0 4
advanced	n 0 42	alienate	b 0 2
advancing	g 0 4	alienated	n 0 6
advertise	b 0 3	align	b 0 2
advertised	d 0 3	aligned	n 0 6
advertised	n 0 6	allege	b 0 1
advertising	g 0 36	alleged	d 0 2
advise	b 0 9	alleged	n 0 8
advised	d 0 17	alleging	g 0 3
advised	n 0 16	alleviate	b 0 5
advises	z 0 2	allocate	b 0 3
advising	g 0 3	allocated	d 0 2
advocate	b 0 4	allocated	n 0 3
advocated	d 0 2	allot	b 0 1
advocated	n 0 2	allotted	d 0 2
advocating	g 0 6	allotted	n 0 8
affect	b 0 34	allow	b 0 72
affected	d 0 4	allowed	d 0 21
affected	n 0 32	allowed	n 0 65
affecting	g 0 5	allowing	g 0 31
affects	z 0 18	allows	z 0 19
affirm	b 0 12	alter	b 0 15
affirmed	d 0 4	altered	d 0 2
affirmed	n 0 2	altered	n 0 20
affirming	g 0 2	altering	g 0 4
afford	b 0 40	amass	b 0 2
afforded	d 0 4	amaze	b 0 3
afforded	n 0 7	amazed	n 0 10
affording	g 0 2	ambush	b 0 2
affords	z 0 5	ambushed	n 0 2
age	b 0 2	amend	b 0 2
aged	n 0 18	amended	d 0 3
aging	g 0 4	amended	n 0 11
agglomerate	b 0 2	amortize	b 0 2
agonize	b 0 0	amount	b 0 30
agonizes	z 0 2	amounted	d 0 2
agree	b 0 51	amounted	n 0 3
agreed	d 0 52	amounts	z 0 20
agreed	n 0 29	amuse	b 0 3
amused	d 0 5		
amused	n 0 4		
analyze	b 0 10		
analyzed	n 0 13		
analysed	n 0 2		
analyzes	z 0 2		
analyzing	g 0 8		
announce	b 0 18		
announced	d 0 53		
announced	n 0 35		
announces	z 0 3		
announcing	g 0 7		
annoy	b 0 2		
annoyed	n 0 6		
answer	b 0 43		
answered	d 0 47		
answered	n 0 20		
answering	g 0 14		
answers	z 0 8		
anticipate	b 0 11		
anticipated	d 0 4		
anticipated	n 0 19		
anticipates	z 0 2		
anticipating	g 0 2		
apologize	b 0 1		
apologized	d 0 4		
appeal	b 0 10		
appealed	d 0 10		
appealed	n 0 3		
appealing	g 0 3		
appear	b 0 117		
appeared	d 0 118		
appeared	n 0 17		
appearing	g 0 16		
appears	z 0 84		
appease	b 0 2		
appeased	n 0 2		
applaud	b 0 5		
applauded	n 0 3		
applauding	g 0 2		
apply	b 0 56		
applied	d 0 22		
applied	n 0 84		
applies	z 0 19		
applying	g 0 29		
appoint	b 0 6		
appointed	d 0 8		
appointed	n 0 34		
appraise	b 0 4		
appreciate	b 0 26		
appreciated	d 0 6		
appreciated	n 0 5		
approach	b 0 15		
approached	d 0 32		
approached	n 0 13		

approaches	z 0 8	ask	b 0 128
approaching	g 0 27	asked	d 0 300
appropriate	b 0 1	asked	n 0 98
appropriated	n 0 10	asking	g 0 67
appropriating	g 0 2	asks	z 0 18
approve	b 0 14	aspire	b 0 3
approved	d 0 12	assail	b 0 3
approved	n 0 28	assailed	n 0 4
approximate	b 0 2	assault	b 0 0
approximated	d 0 2	assaulted	d 0 3
approximated	n 0 3	assaulted	n 0 3
arbitrate	b 0 3	assemble	b 0 9
arch	b 0 1	assembled	d 0 7
arched	d 0 2	assembled	n 0 17
arched	n 0 9	assembling	g 0 6
arches	z 0 2	assent	b 0 0
argue	b 0 29	assented	d 0 2
argued	d 0 17	assert	b 0 19
argued	n 0 12	asserted	d 0 11
argues	z 0 10	asserted	n 0 5
arguing	g 0 10	asserting	g 0 4
arise	b 0 28	asserts	z 0 5
arises	z 0 14	assess	b 0 6
arising	g 0 11	assessed	n 0 9
arose	d 1 18	assessing	g 0 10
arisen	n 1 4	assign	b 0 18
arouse	b 0 5	assigned	d 0 2
aroused	d 0 5	assigned	n 0 51
aroused	n 0 15	assigning	g 0 9
arouses	z 0 2	assigns	z 0 4
arousing	g 0 3	assimilate	b 0 2
arraign	b 0 0	assimilated	n 0 4
arraigned	d 0 2	assist	b 0 22
arrange	b 0 10	assisted	d 0 3
arranged	d 0 11	assisted	n 0 4
arranged	n 0 33	assisting	g 0 7
arranging	g 0 16	associate	b 0 10
arrest	b 0 6	associated	d 0 3
arrested	d 0 4	associated	n 0 58
arrested	n 0 15	associating	g 0 2
arresting	g 0 2	assume	b 0 63
arrive	b 0 24	assumed	d 0 28
arrived	d 0 43	assumed	n 0 44
arrived	n 0 19	assumes	z 0 8
arrives	z 0 7	assuming	g 0 17
arriving	g 0 15	assure	b 0 37
articulate	b 0 2	assured	d 0 16
articulated	n 0 2	assured	n 0 23
ascend	b 0 1	assures	z 0 6
ascended	d 0 2	assuring	g 0 10
ascending	g 0 4	astonish	b 0 0
ascertain	b 0 7	astonished	n 0 5
ascertained	n 0 3	astound	b 0 1
ascribe	b 0 1	astounded	n 0 2
ascribed	n 0 4	attach	b 0 14
attached	d 0 3		
attached	n 0 22		
attaches	z 0 2		
attaching	g 0 3		
attack	b 0 24		
attacked	d 0 12		
attacked	n 0 13		
attacking	g 0 9		
attacks	z 0 5		
attain	b 0 20		
attained	d 0 5		
attained	n 0 3		
attaining	g 0 6		
attempt	b 0 24		
attempted	d 0 18		
attempted	n 0 15		
attempting	g 0 23		
attempts	z 0 7		
attend	b 0 54		
attended	d 0 24		
attended	n 0 12		
attending	g 0 23		
attends	z 0 6		
attest	b 0 2		
attested	d 0 2		
attested	n 0 2		
attract	b 0 19		
attracted	d 0 11		
attracted	n 0 14		
attracting	g 0 4		
attracts	z 0 3		
attribute	b 0 2		
attributed	d 0 6		
attributed	n 0 12		
attributing	g 0 3		
augment	b 0 2		
augmented	d 0 4		
augmented	n 0 5		
authorize	b 0 5		
authorized	d 0 4		
authorized	n 0 33		
authorizes	z 0 2		
authorizing	g 0 5		
avail	b 0 3		
availed	d 0 2		
avenge	b 0 2		
avenging	g 0 2		
average	b 0 5		
averaged	d 0 13		
averaging	g 0 8		
avert	b 0 1		
averted	d 0 2		
averting	g 0 3		
avoid	b 0 58		
avoided	d 0 7		

avoided	n	0	12
avoiding	g	0	11
avoids	z	0	3
await	b	0	9
awaited	d	0	6
awaiting	g	0	7
awaits	z	0	3
awake	b	0	2
awoke	d	1	9
awaken	b	0	7
awakened	n	0	3
award	b	0	3
awarded	d	0	2
awarded	n	0	15
awarding	g	0	2
babble	b	0	0
babbled	d	0	2
back	b	0	25
backed	d	0	21
backed	n	0	3
backing	g	0	5
backs	z	0	3
backstitch	b	0	2
bake	b	0	3
baked	n	0	7
baking	g	0	3
balance	b	0	8
balanced	n	0	21
balancing	g	0	2
balk	b	0	0
balked	d	0	2
bandage	b	0	0
bandaged	n	0	4
bang	b	0	2
banged	d	0	4
banging	g	0	3
banish	b	0	4
banished	n	0	6
bank	b	0	0
banked	n	0	3
banking	g	0	2
bankrupt	b	0	2
ban	b	0	1
banned	n	0	2
bar	b	0	3
barred	d	0	2
barred	n	0	6
barring	g	0	3
bars	z	0	3
barge	b	0	2
barging	g	0	2
base	b	0	3
based	d	0	3
based	n	0	116
bases	z	0	4
basing	g	0	4
basting	g	0	2
bat	b	0	5
batting	g	0	15
bathe	b	0	4
bathed	n	0	6
bathing	g	0	15
batter	b	0	0
battered	d	0	2
battered	n	0	7
battering	g	0	2
battle	b	0	3
battling	g	0	3
bawl	b	0	0
bawled	d	0	2
bay	b	0	0
bayed	d	0	2
bear	b	0	43
bearing	g	0	17
bears	z	0	17
bore	d	1	14
born	n	1	112
beat	b	0	25
beating	g	0	10
beats	z	0	3
beat	d	1	12
beaten	n	1	15
beckon	b	0	1
beckoned	d	0	6
beckons	z	0	3
become	b	0	235
becomes	z	0	104
becoming	g	0	54
became	d	1	246
become	n	1	124
befall	b	0	2
befell	d	1	1
beg	b	0	11
begged	d	0	13
begging	g	0	9
begin	b	0	84
beginning	g	0	81
begins	z	0	55
begrudge	b	0	2
behave	b	0	13
behaved	d	0	10
behaved	n	0	3
behaves	z	0	2
behaving	g	0	4
behold	b	0	4
belch	b	0	0
belched	d	0	4
belie	b	0	0
belied	d	0	3
believe	b	0	200
believed	d	0	52
believed	n	0	25
believes	z	0	43
believing	g	0	14
bellow	b	0	0
bellowed	d	0	6
bellowing	g	0	2
belong	b	0	37
belonged	d	0	14
belonged	n	0	3
belonging	g	0	12
belongs	z	0	22
belt	b	0	0
belted	d	0	2
bend	b	0	12
bending	g	0	6
bent	d	1	14
bent	n	1	17
benefit	b	0	20
benefited	n	0	3
bestow	b	0	2
bestowed	d	0	3
bestowed	n	0	4
bet	b	0	13
betting	g	0	4
bet	d	1	1
betide	b	0	2
betray	b	0	4
betrayed	d	0	6
betrayed	n	0	2
betrays	z	0	3
better	b	0	6
bevel	b	0	2
beveled	n	0	3
beware	b	0	3
bewitch	b	0	0
bewitched	n	0	2
bicker	b	0	0
bickering	g	0	2
bid	b	0	7
bid	d	1	1
bid	n	1	1
bind	b	0	2
binding	g	0	10
binds	z	0	2
bound	d	1	5
bound	n	1	32
birdie	b	0	1
birdied	d	0	2
birdied	n	0	2
blame	b	0	23
blamed	d	0	5
blamed	n	0	2
blaming	g	0	2
blast	b	0	3

blasted	d 0 3	boarded	n 0 3
blasting	g 0 2	boarding	g 0 4
blaze	b 0 3	boast	b 0 7
blazed	d 0 2	boasted	d 0 5
blazing	g 0 6	boasting	g 0 3
bleached	n 0 5	boasts	z 0 2
bleaching	g 0 2	bobb	b 0 0
bleed	b 0 2	bobbed	d 0 2
bleeding	g 0 13	bobbing	g 0 3
bled	d 1 2	bogey	b 0 0
bled	n 1 1	bogeyed	d 0 2
blend	b 0 4	boil	b 0 7
blended	d 0 2	boiled	n 0 9
blended	n 0 2	boiling	g 0 9
bless	b 0 9	bolster	b 0 2
blessed	n 0 13	bolt	b 0 2
blind	b 0 1	bolted	d 0 3
blinded	n 0 4	bolted	n 0 4
blinding	g 0 2	boost	b 0 5
blindfold	b 0 0	boosted	d 0 2
blindfolded	n 0 2	boosting	g 0 3
blink	b 0 4	boot	b 0 3
blinked	d 0 6	border	b 0 0
blinking	g 0 3	bordered	n 0 2
blister	b 0 1	bordering	g 0 5
blistered	n 0 2	borders	z 0 2
block	b 0 5	bore	b 0 7
blocked	d 0 5	bored	d 0 3
blocked	n 0 7	bored	n 0 11
blocking	g 0 3	boring	g 0 4
blockade	b 0 0	borrow	b 0 9
blockading	g 0 2	borrowed	d 0 5
bloom	b 0 3	borrowed	n 0 9
bloomed	d 0 6	borrowing	g 0 7
blooming	g 0 8	bother	b 0 22
blossom	b 0 3	bothered	d 0 7
blot	b 0 1	bothered	n 0 7
blotted	n 0 2	bothering	g 0 6
blotting	g 0 2	bothers	z 0 3
blow	b 0 8	bottle	b 0 0
blowing	g 0 18	bottled	n 0 3
blows	z 0 5	bounce	b 0 4
blew	d 1 12	bounced	d 0 13
blown	n 1 9	bounced	n 0 3
blunder	b 0 0	bouncing	g 0 8
blundered	n 0 2	bound	b 0 2
blunt	b 0 2	bounded	d 0 2
blurt	b 0 0	bounded	n 0 7
blurted	d 0 2	bounding	g 0 2
blush	b 0 2	bow	b 0 3
blushed	d 0 4	bowed	d 0 6
blushed	n 0 2	bowing	g 0 2
blushing	g 0 4	box	b 0 2
board	b 0 5	boxed	n 0 2
boarded	d 0 2	brace	b 0 2

braced	n 0 4	breakfast	b 0 0
brag	b 0 1	breakfasted	d 0 2
bragged	d 0 2	breathe	b 0 7
branch	b 0 2	breathed	d 0 9
branched	n 0 2	breathes	z 0 2
brand	b 0 0	breathing	g 0 13
branded	n 0 2	breed	b 0 3
break	b 0 65	breeding	g 0 2
breaking	g 0 21	bribe	b 0 1
breaks	z 0 9	bribed	d 0 2
broke	d 1 67	bridge	b 0 4
broken	n 1 63	brighten	b 0 0
		brightened	d 0 2
		bring	b 0 158
		bringing	g 0 36
		brings	z 0 40
		brought	d 1 134
		brought	n 1 119
		bristle	b 0 1
		bristled	d 0 3
		bristling	g 0 3
		broaden	b 0 8
		broadened	d 0 3
		broadened	n 0 4
		broadening	g 0 5
		broadens	z 0 2
		broil	b 0 1
		broiled	n 0 2
		bruise	b 0 1
		bruised	n 0 7
		bruising	g 0 2
		brush	b 0 13
		brushed	d 0 14
		brushed	n 0 6
		brushing	g 0 5
		buckle	b 0 3
		bud	b 0 2
		budge	b 0 3
		budget	b 0 2
		budgeted	n 0 2
		budgeting	g 0 4
		build	b 0 84
		building	g 0 54

builds	z 0 7	canned	n 0 6
built	d 1 21	canning	g 0 6
built	n 1 82	cancel	b 0 7
bulge	b 0 0	canceled	n 0 5
bulged	d 0 3	cap	b 0 3
bulging	g 0 3	capitalize	b 0 4
bump	b 0 4	capitalizing	g 0 2
bumped	d 0 2	capitulate	b 0 0
bumping	g 0 2	capitulated	d 0 2
bury	b 0 5	capture	b 0 13
buried	d 0 3	captured	d 0 2
buried	n 0 15	captured	n 0 15
burn	b 0 10	capturing	g 0 2
burned	d 0 15	care	b 0 75
burned	n 0 25	cared	d 0 9
burning	g 0 44	cared	n 0 6
burns	z 0 2	cares	z 0 8
burnt	n 1 6	caring	g 0 10
burst	b 0 9	caress	b 0 1
burst	d 1 11	caressed	d 0 3
burst	n 1 3	caressing	g 0 5
bursting	g 0 13	carry	b 0 88
butt	b 0 2	carried	d 0 60
butted	n 0 3	carried	n 0 65
buy	b 0 69	carries	z 0 22
buying	g 0 24	carrying	g 0 69
buys	z 0 11	carve	b 0 3
bought	d 1 32	carved	n 0 14
bought	n 1 24	carving	g 0 6
buzz	b 0 1	case	b 0 2
buzzed	d 0 2	cash	b 0 2
buzzing	g 0 6	cast	b 0 6
bypass	b 0 2	casting	g 0 3
cable	b 0 2	casts	z 0 3
cackle	b 0 0	cast	d 1 4
cackled	d 0 3	cast	n 1 12
calculate	b 0 4	catalogue	b 0 0
calculated	d 0 2	catalogued	n 0 3
calculated	n 0 33	catch	b 0 39
calculating	g 0 7	catching	g 0 7
call	b 0 134	caught	d 1 54
called	d 0 165	caught	n 1 44
called	n 0 236	cater	b 0 3
calling	g 0 44	catering	g 0 3
calls	z 0 47	cause	b 0 52
calm	b 0 6	caused	d 0 39
calmed	d 0 3	caused	n 0 51
calmed	n 0 3	causes	z 0 27
calming	g 0 2	causing	g 0 17
calve	b 0 0	caution	b 0 1
calving	g 0 3	cautioned	d 0 6
campaign	b 0 5	cease	b 0 15
campaigned	d 0 4	ceased	d 0 8
campaigning	g 0 3	ceased	n 0 4
can	b 0 2	ceases	z 0 3
ceasing	g 0 2		
celebrate	b 0 4		
celebrated	d 0 5		
celebrated	n 0 9		
celebrates	z 0 2		
celebrating	g 0 5		
center	b 0 9		
centered	d 0 5		
centered	n 0 9		
centering	g 0 4		
centers	z 0 7		
centralize	b 0 0		
centralized	n 0 8		
centralizing	g 0 2		
certify	b 0 5		
certified	n 0 7		
challenge	b 0 14		
challenged	d 0 4		
challenged	n 0 5		
challenges	z 0 3		
challenging	g 0 3		
chance	b 0 1		
chanced	d 0 2		
change	b 0 77		
changed	d 0 26		
changed	n 0 69		
changes	z 0 10		
changing	g 0 43		
channel	b 0 1		
channeled	n 0 3		
chant	b 0 1		
chanted	d 0 3		
chanted	n 0 3		
chanting	g 0 2		
characterize	b 0 6		
characterized	d 0 6		
characterized	n 0 15		
characterizes	z 0 4		
charge	b 0 15		
charged	d 0 17		
charged	n 0 40		
charging	g 0 8		
charm	b 0 2		
charmed	n 0 2		
chart	b 0 1		
charted	n 0 6		
charting	g 0 4		
chase	b 0 4		
chasing	g 0 2		
chat	b 0 2		
chatted	d 0 2		
chattered	d 0 3		
chattering	g 0 4		
chatting	g 0 2		
cheat	b 0 2		

cheated	n 0 4	claiming	g 0 16
check	b 0 51	claims	z 0 19
checked	d 0 10	clamber	b 0 0
checked	n 0 21	clambered	d 0 6
checking	g 0 4	clamp	b 0 0
cheer	b 0 2	clamped	d 0 6
cherish	b 0 5	clamped	n 0 2
cherished	d 0 4	clamping	g 0 3
cherished	n 0 12	clap	b 0 0
cherishing	g 0 2	clapped	d 0 4
chew	b 0 2	clapping	g 0 4
chewed	d 0 4	clarify	b 0 13
chewing	g 0 10	clarified	n 0 7
chide	b 0 2	clarifying	g 0 3
chill	b 0 0	clasp	b 0 0
chilled	d 0 2	clasped	d 0 2
chilled	n 0 5	clasping	g 0 4
chilling	g 0 5	classify	b 0 6
chin	b 0 2	classified	n 0 14
chinning	g 0 2	clatter	b 0 1
chip	b 0 0	clattered	d 0 5
chipped	n 0 3	clean	b 0 19
chipping	g 0 6	cleaned	d 0 3
choke	b 0 9	cleaned	n 0 13
choked	d 0 4	cleaning	g 0 22
choked	n 0 3	clear	b 0 14
choking	g 0 6	cleared	d 0 13
choose	b 0 50	cleared	n 0 10
chooses	z 0 8	clearing	g 0 10
choosing	g 0 11	clench	b 0 1
chose	d 1 37	clenched	n 0 4
chosen	n 1 71	click	b 0 0
chop	b 0 1	clicked	d 0 5
chopped	n 0 2	clicked	n 0 3
chopping	g 0 5	climb	b 0 11
chortle	b 0 0	climbed	d 0 41
chortled	d 0 3	climbed	n 0 3
chuck	b 0 3	climbing	g 0 10
chuckle	b 0 2	clinch	b 0 2
chuckled	d 0 8	cling	b 0 6
circle	b 0 2	clinging	g 0 7
circled	d 0 9	clings	z 0 3
circling	g 0 2	clung	d 1 13
circulate	b 0 2	clung	n 1 1
circulated	d 0 2	clip	b 0 0
circulated	n 0 2	clipped	d 0 2
circulating	g 0 5	clog	b 0 2
cite	b 0 7	clogging	g 0 2
cited	d 0 11	close	b 0 39
cited	n 0 13	closed	d 0 39
cites	z 0 10	closed	n 0 67
citing	g 0 3	closes	z 0 6
claim	b 0 29	closing	g 0 23
claimed	d 0 25	cluster	b 0 5
claimed	n 0 10	clustered	n 0 3
clutch	b 0 1		
clutched	d 0 5		
clutched	n 0 2		
clutching	g 0 8		
clutter	b 0 0		
cluttered	n 0 2		
cooperate	b 0 4		
coast	b 0 0		
coasted	d 0 2		
cock	b 0 0		
cocked	d 0 4		
cocked	n 0 2		
coddle	b 0 0		
coddled	n 0 2		
coerce	b 0 2		
coincide	b 0 12		
coincided	d 0 6		
coincides	z 0 5		
coin	b 0 1		
coined	n 0 2		
collaborate	b 0 2		
collaborated	d 0 5		
collaborated	n 0 4		
collapse	b 0 1		
collapsed	d 0 10		
collapsed	n 0 3		
collapsing	g 0 3		
collar	b 0 3		
collect	b 0 16		
collected	d 0 7		
collected	n 0 37		
collecting	g 0 13		
collects	z 0 5		
color	b 0 5		
colored	n 0 31		
coloring	g 0 4		
colors	z 0 3		
combat	b 0 4		
comb	b 0 0		
combed	d 0 3		
combine	b 0 15		
combined	d 0 6		
combined	n 0 34		
combines	z 0 7		
combining	g 0 10		
come	b 0 434		
comes	z 0 137		
coming	g 0 161		
came	d 1 622		
come	n 1 191		
comfort	b 0 2		
comforting	g 0 8		
command	b 0 10		
commanded	d 0 10		
commanded	n 0 5		

commanding	g 0 10	comply	b 0 5
commands	z 0 3	complied	d 0 2
commemorate	b 0 2	complied	n 0 4
commemorated	n 0 2	complying	g 0 3
commence	b 0 2	compose	b 0 6
commenced	d 0 6	composed	d 0 4
commencing	g 0 8	composed	n 0 36
commend	b 0 7	composes	z 0 2
commended	n 0 3	composing	g 0 2
commending	g 0 2	compound	b 0 2
comment	b 0 7	compounded	n 0 12
commented	d 0 16	comprehend	b 0 5
commented	n 0 2	comprehending	g 0 3
commenting	g 0 5	compress	b 0 2
commit	b 0 15	compressed	n 0 8
commits	z 0 2	comprise	b 0 11
committed	d 0 4	comprised	d 0 3
committed	n 0 24	comprised	n 0 5
committing	g 0 5	comprises	z 0 3
communicate	b 0 13	comprising	g 0 3
communicated	n 0 3	compromise	b 0 3
communicating	g 0 6	compromising	g 0 3
commute	b 0 10	compute	b 0 7
commutes	z 0 2	computed	n 0 21
commuting	g 0 5	computes	z 0 2
compare	b 0 28	computing	g 0 11
compared	d 0 10	conceal	b 0 7
compared	n 0 61	concealed	n 0 7
compares	z 0 6	conceals	z 0 2
comparing	g 0 9	concede	b 0 8
compel	b 0 4	conceded	d 0 5
compelled	n 0 17	conceded	n 0 6
compels	z 0 2	conceding	g 0 3
compensate	b 0 3	conceive	b 0 14
compensated	n 0 4	conceived	d 0 7
compensating	g 0 2	conceived	n 0 20
compete	b 0 23	conceives	z 0 2
competed	d 0 2	conceiving	g 0 2
competing	g 0 15	concentrate	b 0 10
compile	b 0 1	concentrated	n 0 29
compiled	d 0 2	concentrates	z 0 2
compiled	n 0 8	concentrating	g 0 7
compiling	g 0 4	concern	b 0 12
complain	b 0 11	concerned	d 0 4
complained	d 0 21	concerned	n 0 131
complaining	g 0 5	concerns	z 0 14
complains	z 0 3	conclude	b 0 16
complement	b 0 2	concluded	d 0 21
complete	b 0 19	concluded	n 0 11
completed	d 0 6	concludes	z 0 4
completed	n 0 63	concluding	g 0 8
completes	z 0 6	concur	b 0 4
completing	g 0 13	concurred	d 0 2
complicate	b 0 2	concurs	z 0 3
complicated	n 0 29	condemn	b 0 4
condemned	d 0 6
condemned	n 0 13
condemning	g 0 4
condemns	z 0 3
condense	b 0 1
condensed	n 0 9
condition	b 0 1
conditioned	n 0 19
conditioning	g 0 3
conduct	b 0 20
conducted	d 0 14
conducted	n 0 41
conducting	g 0 13
conducts	z 0 3
confer	b 0 3
conferred	n 0 4
confess	b 0 11
confessed	d 0 6
confesses	z 0 3
confessing	g 0 3
confide	b 0 3
confided	d 0 7
confiding	g 0 2
confine	b 0 2
confined	n 0 15
confining	g 0 3
confirm	b 0 16
confirmed	d 0 8
confirmed	n 0 12
confirming	g 0 2
confirms	z 0 3
confiscate	b 0 0
confiscated	n 0 2
conflict	b 0 3
conflicting	g 0 8
conform	b 0 10
conformed	n 0 2
conforms	z 0 5
confront	b 0 8
confronted	d 0 5
confronted	n 0 27
confronting	g 0 10
confronts	z 0 5
confuse	b 0 5
confused	d 0 4
confused	n 0 40
congeal	b 0 0
congealed	d 0 2
congealed	n 0 2
congest	b 0 0
congested	n 0 2
congratulate	b 0 4
congratulated	n 0 3
congregate	b 0 2
conjure	b 0 1

conjures	z 0 2	contained	d 0 35
connect	b 0 3	contained	n 0 25
connected	d 0 4	containing	g 0 45
connected	n 0 29	contains	z 0 38
connecting	g 0 6	contemplate	b 0 7
connects	z 0 2	contemplated	n 0 5
conquer	b 0 4	contemplating	g 0 6
conquered	n 0 2	contend	b 0 6
conquering	g 0 3	contended	d 0 6
consent	b 0 2	contended	n 0 6
consented	d 0 2	contends	z 0 5
consented	n 0 2	content	b 0 3
conserve	b 0 3	contented	n 0 8
conserving	g 0 2	contest	b 0 2
consider	b 0 127	contested	n 0 2
considered	d 0 31	continue	b 0 107
considered	n 0 120	continued	d 0 83
considering	g 0 24	continued	n 0 50
considers	z 0 15	continues	z 0 41
consign	b 0 2	continuing	g 0 61
consist	b 0 17	contract	b 0 6
consisted	d 0 22	contracted	d 0 4
consisted	n 0 2	contracted	n 0 4
consisting	g 0 25	contracting	g 0 2
consists	z 0 43	contracts	z 0 5
consolidate	b 0 2	contradict	b 0 4
consolidated	n 0 6	contradicts	z 0 2
consolidating	g 0 2	contrast	b 0 5
conspire	b 0 1	contrasted	n 0 4
conspired	d 0 2	contrasting	g 0 11
constitute	b 0 29	contrasts	z 0 2
constituted	d 0 8	contribute	b 0 44
constituted	n 0 3	contributed	d 0 24
constitutes	z 0 11	contributed	n 0 15
constituting	g 0 3	contributes	z 0 10
constrain	b 0 0	contributing	g 0 15
constrained	n 0 2	contrive	b 0 1
construct	b 0 12	contrived	n 0 3
constructed	d 0 2	control	b 0 28
constructed	n 0 35	controlled	d 0 5
constructing	g 0 7	controlled	n 0 34
construe	b 0 1	controlling	g 0 23
construed	n 0 5	controls	z 0 5
consult	b 0 11	convene	b 0 0
consulted	d 0 6	convened	d 0 2
consulted	n 0 11	converge	b 0 3
consulting	g 0 13	converse	b 0 3
consume	b 0 2	convert	b 0 9
consumed	n 0 12	converted	d 0 2
consuming	g 0 5	converted	n 0 18
contact	b 0 8	converting	g 0 2
contacted	d 0 2	convey	b 0 13
contacted	n 0 2	conveyed	d 0 3
contacting	g 0 2	conveyed	n 0 6
contain	b 0 45	conveys	z 0 4
convict	b 0 1		
convicted	n 0 13		
convince	b 0 4		
convinced	d 0 7		
convinced	n 0 43		
convincing	g 0 3		
cook	b 0 14		
cooked	d 0 2		
cooked	n 0 8		
cooking	g 0 26		
cool	b 0 7		
cooled	d 0 3		
cooled	n 0 14		
cooling	g 0 33		
cools	z 0 2		
cooperate	b 0 11		
cooperated	n 0 2		
cooperating	g 0 7		
coordinate	b 0 7		
coordinated	n 0 14		
coordinating	g 0 4		
cope	b 0 21		
coping	g 0 8		
copy	b 0 2		
copied	n 0 2		
core	b 0 2		
cork	b 0 0		
corked	n 0 2		
correct	b 0 12		
corrected	d 0 3		
corrected	n 0 6		
correlate	b 0 3		
correlated	n 0 3		
correlating	g 0 2		
correspond	b 0 7		
corresponded	d 0 4		
corresponding	g 0 9		
corresponds	z 0 6		
corroborate	b 0 2		
corrupt	b 0 1		
corrupted	n 0 2		
corrupting	g 0 2		
cost	b 0 29		
costing	g 0 5		
costs	z 0 12		
cost	d 1 10		
cost	n 1 5		
cough	b 0 4		
coughed	d 0 2		
coughing	g 0 2		
counsel	b 0 1		
counseled	d 0 2		
counseling	g 0 5		
count	b 0 26		
counted	d 0 11		

counted	n 0 6	cringing	g 0 3
counting	g 0 12	cripple	b 0 0
counts	z 0 6	crippled	n 0 6
counter	b 0 2	crippling	g 0 6
countered	d 0 2	criticize	b 0 4
counteract	b 0 4	criticized	d 0 3
counteracting	g 0 2	criticized	n 0 11
couple	b 0 0	criticizing	g 0 2
coupled	d 0 3	croon	b 0 0
coupled	n 0 11	crooned	d 0 2
coupling	g 0 7	cross	b 0 25
court	b 0 2	crossed	d 0 26
courting	g 0 2	crossed	n 0 16
cover	b 0 53	crosses	z 0 3
covered	d 0 14	crossing	g 0 14
covered	n 0 90	crouch	b 0 2
covering	g 0 30	crouched	d 0 10
covers	z 0 15	crouched	n 0 6
covet	b 0 1	crouching	g 0 3
coveted	n 0 4	crowd	b 0 2
crack	b 0 9	crowded	d 0 8
cracked	d 0 11	crowded	n 0 24
cracked	n 0 6	crowding	g 0 5
cracking	g 0 14	crow	b 0 0
crash	b 0 4	crowed	d 0 2
crashed	d 0 7	crowing	g 0 2
crashed	n 0 5	crown	b 0 1
crashing	g 0 7	crowned	n 0 7
crave	b 0 2	crowning	g 0 3
crawl	b 0 9	crumble	b 0 2
crawled	d 0 17	crumbled	n 0 2
crawled	n 0 3	crumbling	g 0 2
crawling	g 0 8	crush	b 0 1
craze	b 0 0	crushed	d 0 2
crazed	n 0 2	crushed	n 0 8
creak	b 0 0	crushing	g 0 6
creaked	d 0 6	cry	b 0 18
creaking	g 0 4	cried	d 0 25
crease	b 0 0	cried	n 0 5
creased	n 0 2	crying	g 0 15
create	b 0 54	culminate	b 0 2
created	d 0 18	culminated	d 0 2
created	n 0 63	culminates	z 0 5
creates	z 0 13	culminating	g 0 2
creating	g 0 29	cultivate	b 0 3
credit	b 0 2	cultivated	d 0 2
credited	d 0 2	cultivated	n 0 8
credited	n 0 10	cultivating	g 0 2
credits	z 0 2	cup	b 0 1
creep	b 0 7	cupped	d 0 2
creeping	g 0 8	cupped	n 0 2
crept	d 1 9	cure	b 0 12
crept	n 1 2	cured	n 0 6
cringe	b 0 0	curl	b 0 2
cringed	d 0 2	curled	d 0 6
curled	n 0 7		
curling	g 0 2		
curse	b 0 4		
cursed	d 0 7		
cursed	n 0 4		
cursing	g 0 8		
curtail	b 0 4		
curtailed	n 0 2		
curve	b 0 0		
curved	n 0 6		
curving	g 0 4		
cushion	b 0 2		
cushioning	g 0 2		
cut	b 0 88		
cuts	z 0 14		
cutting	g 0 58		
cut	d 1 25		
cut	n 1 60		
damage	b 0 5		
damaged	d 0 5		
damaged	n 0 2		
damages	z 0 2		
damaging	g 0 3		
damn	b 0 10		
damned	n 0 19		
dampen	b 0 2		
dampened	n 0 2		
dance	b 0 17		
danced	d 0 8		
danced	n 0 2		
dances	z 0 2		
dancing	g 0 30		
dangle	b 0 1		
dangled	d 0 2		
dangling	g 0 4		
dare	b 0 16		
dared	d 0 7		
dared	n 0 7		
dares	z 0 3		
daring	g 0 11		
darken	b 0 0		
darkened	d 0 5		
darkened	n 0 2		
darkening	g 0 4		
darn	b 0 2		
darned	n 0 3		
dart	b 0 0		
darted	d 0 6		
dash	b 0 1		
dashed	d 0 4		
dashed	n 0 4		
dashing	g 0 4		
date	b 0 5		
dated	n 0 19		
dates	z 0 8		

dating	g 0 4	deduced	d 0 2
dazzle	b 0 1	deduced	n 0 4
dazzled	n 0 2	deduct	b 0 12
dazzling	g 0 9	deducted	n 0 3
deal	b 0 45	deem	b 0 1
dealing	g 0 43	deemed	d 0 3
deals	z 0 14	deemed	n 0 12
dealt	d 1 8	defeat	b 0 7
dealt	n 1 14	defeated	d 0 5
debate	b 0 2	defeated	n 0 10
debated	n 0 4	defeating	g 0 3
debating	g 0 3	defend	b 0 21
decant	b 0 0	defended	d 0 9
decanted	n 0 2	defended	n 0 9
decanting	g 0 3	defending	g 0 13
decay	b 0 2	defends	z 0 4
decayed	n 0 3	defy	b 0 7
decaying	g 0 4	defying	g 0 2
deceive	b 0 1	defied	d 0 2
deceived	n 0 4	defied	n 0 2
decide	b 0 40	define	b 0 27
decided	d 0 105	defined	n 0 38
decided	n 0 36	defines	z 0 5
decides	z 0 12	defining	g 0 10
deciding	g 0 12	defraud	b 0 2
declaim	b 0 0	defray	b 0 2
declaimed	d 0 2	delay	b 0 8
declare	b 0 8	delayed	d 0 6
declared	d 0 52	delayed	n 0 19
declared	n 0 14	delegate	b 0 4
declares	z 0 11	delegated	n 0 4
declaring	g 0 10	delegating	g 0 2
decline	b 0 7	delight	b 0 2
declined	d 0 15	delighted	n 0 15
declined	n 0 2	deliver	b 0 18
declines	z 0 4	delivered	d 0 13
declining	g 0 9	delivered	n 0 24
decompose	b 0 1	delivering	g 0 9
decomposes	z 0 2	delivers	z 0 6
decomposing	g 0 2	delude	b 0 2
decorate	b 0 2	deluded	n 0 2
decorated	n 0 5	demand	b 0 22
decorating	g 0 4	demanded	d 0 33
decrease	b 0 10	demanded	n 0 9
decreased	d 0 6	demanding	g 0 16
decreased	n 0 2	demands	z 0 12
decreases	z 0 7	democratize	b 0 3
decreasing	g 0 6	demonstrate	b 0 28
decry	b 0 2	demonstrated	d 0 9
decried	d 0 2	demonstrated	n 0 24
dedicate	b 0 0	demonstrates	z 0 6
dedicated	d 0 2	demonstrating	g 0 6
dedicated	n 0 21	demoralize	b 0 3
dedicates	z 0 2	demoralizes	z 0 2
deduce	b 0 3	deny	b 0 47

denying	g 0 9	depart	b 0 7
denied	d 0 10	departed	d 0 5
denied	n 0 37	departed	n 0 4
denies	z 0 6	departing	g 0 10
denote	b 0 4	departs	z 0 2
denoted	d 0 2	depend	b 0 45
denoted	n 0 7	depended	d 0 9
denotes	z 0 7	depending	g 0 3
denoting	g 0 5	depends	z 0 49
denounce	b 0 5	depict	b 0 3
denounced	d 0 2	depicted	d 0 2
denounced	n 0 5	depicted	n 0 6
denouncing	g 0 4	depicting	g 0 6
		deplore	b 0 1
		deplored	d 0 2
		deplores	z 0 3
		deprive	b 0 3
		deprived	n 0 7
		depriving	g 0 3
		derive	b 0 13
		derived	n 0 38
		derives	z 0 9
		deriving	g 0 4
		descend	b 0 4
		descended	d 0 4
		descended	n 0 4
		descending	g 0 10
		descends	z 0 2
		describe	b 0 41
		described	d 0 28
		described	n 0 92
		describes	z 0 22
		describing	g 0 17
		desert	b 0 3
		deserted	n 0 14
		deserts	z 0 2
		deserve	b 0 12
		deserved	d 0 10
		deserved	n 0 2
		deserves	z 0 16
		design	b 0 4
		designed	d 0 9

designed	n 0 99	dictates	z 0 4
designing	g 0 8	dictating	g 0 2
designs	z 0 2	die	b 0 57
designate	b 0 3	died	d 0 63
designated	d 0 4	died	n 0 23
designated	n 0 13	dies	z 0 9
designating	g 0 3	differ	b 0 18
desire	b 0 11	differed	d 0 12
desired	d 0 9	differs	z 0 10
desired	n 0 41	differentiate	b 0 2
desires	z 0 4	differentiated	n 0 4
desiring	g 0 5	diffuse	b 0 1
despise	b 0 7	diffused	n 0 2
despised	d 0 2	diffusing	g 0 3
destroy	b 0 48	dig	b 0 9
destroyed	d 0 8	digging	g 0 7
destroyed	n 0 31	dug	d 1 7
destroying	g 0 17	dug	n 1 8
detach	b 0 1	digest	b 0 0
detached	n 0 11	digesting	g 0 2
detailed	n 0 52	dilate	b 0 2
detect	b 0 10	dilated	n 0 2
detected	n 0 12	dilute	b 0 0
detecting	g 0 6	diluted	n 0 6
deteriorate	b 0 1	diluting	g 0 3
deteriorated	n 0 3	diminish	b 0 3
deteriorating	g 0 2	diminished	n 0 9
determine	b 0 107	diminishes	z 0 3
determined	d 0 7	diminishing	g 0 8
determined	n 0 112	dine	b 0 2
determines	z 0 14	dined	d 0 2
determining	g 0 33	dining	g 0 26
detest	b 0 1	dip	b 0 2
detested	d 0 2	dipped	d 0 3
devastate	b 0 1	direct	b 0 14
devastated	n 0 2	directed	d 0 10
devastating	g 0 5	directed	n 0 58
develop	b 0 89	directing	g 0 7
developed	d 0 43	directs	z 0 5
developed	n 0 127	disable	b 0 1
developing	g 0 52	disabled	n 0 10
develops	z 0 11	disabling	g 0 3
devise	b 0 8	disabuse	b 0 2
devised	d 0 2	disagree	b 0 7
devised	n 0 14	disagreed	d 0 3
devote	b 0 15	disagrees	z 0 2
devoted	d 0 14	disappear	b 0 11
devoted	n 0 37	disappeared	d 0 21
devoting	g 0 10	disappeared	n 0 14
devour	b 0 2	disappearing	g 0 5
diagnose	b 0 3	disappears	z 0 3
dial	b 0 0	disapprove	b 0 4
dialed	d 0 2	disapproved	d 0 3
dictate	b 0 3	disarm	b 0 2
dictated	n 0 4	disarmed	n 0 3
disarming	g 0 3
discern	b 0 4
discerned	n 0 2
discerning	g 0 2
discharge	b 0 3
discharged	n 0 8
discharging	g 0 3
discipline	b 0 2
disciplined	d 0 3
disciplined	n 0 8
disclose	b 0 9
disclosed	d 0 7
disclosed	n 0 7
disconnect	b 0 0
disconnected	n 0 4
discontinue	b 0 2
discontinued	d 0 2
discontinued	n 0 5
discount	b 0 4
discounted	d 0 2
discourage	b 0 9
discouraged	n 0 14
discover	b 0 40
discovered	d 0 30
discovered	n 0 43
discovering	g 0 7
discovers	z 0 3
discuss	b 0 28
discussed	d 0 18
discussed	n 0 47
discusses	z 0 4
discussing	g 0 16
disdain	b 0 1
disdaining	g 0 2
disentangle	b 0 2
disfigure	b 0 0
disfigured	d 0 2
disfigured	n 0 3
disguise	b 0 3
disguised	d 0 2
disguised	n 0 9
disgust	b 0 0
disgusted	n 0 6
dishearten	b 0 2
disintegrate	b 0 2
dislike	b 0 7
disliked	d 0 11
dislikes	z 0 3
dislodge	b 0 2
dismember	b 0 0
dismembered	d 0 2
dismiss	b 0 5
dismissed	d 0 7
dismissed	n 0 7
dismissing	g 0 3

dismount b 0 0
dismounted d 0 3
dismounted n 0 2
dismounting g 0 2
disobey b 0 0
disobeyed n 0 3
dispatch b 0 2
dispatched d 0 2
dispatched n 0 3
dispatching g 0 3
dispel b 0 3
dispelled n 0 7
dispense b 0 4
dispensed n 0 2
disperse b 0 2
dispersed n 0 6
displace b 0 3
displaced n 0 2
display b 0 12
displayed d 0 8
displayed n 0 13
displaying g 0 6
displays z 0 6
dispose b 0 5
disposed d 0 4
disposed n 0 14
disprove b 0 3
dispute b 0 4
disputed n 0 2
disregard b 0 2
disregarded d 0 3
disregarding g 0 3
disrupt b 0 5
disrupted d 0 2
disrupted n 0 3
disrupting g 0 2
dissolve b 0 5
dissolved n 0 15
dissolving g 0 3
dissuade b 0 3
distinguish b 0 19
distinguished n 0 40
distinguishes z 0 5
distinguishing g 0 6
distort b 0 4
distorted n 0 10
distract b 0 2
distracted d 0 3
distracted n 0 2
distribute b 0 6
distributed d 0 2
distributed n 0 25
distributes z 0 2
distributing g 0 4
distrust b 0 2
disturb b 0 10
disturbed d 0 4
disturbed n 0 22
disturbing g 0 2
dive b 0 3
dived d 0 4
diving g 0 3
divert b 0 1
diverted n 0 3
diverting g 0 3
divide b 0 14
divided d 0 11
divided n 0 45
divides z 0 6
dividing g 0 7
divorce b 0 6
divorced d 0 2
divorced n 0 6
dock b 0 2
document b 0 2
documented n 0 5
dodge b 0 4
dodged d 0 2
dodging g 0 2
do b 0 0
doing g 0 159
done n 1 315
dominate b 0 8
dominated d 0 5
dominated n 0 15
dominates z 0 7
dominating g 0 2
donate b 0 3
donated n 0 6
don b 0 0
donned d 0 3
doom b 0 0
doomed d 0 2
doomed n 0 8
dot b 0 2
doting g 0 2
dotted n 0 2
double b 0 5
doubled d 0 4
doubled n 0 7
doubles z 0 2
doubling g 0 7
doubt b 0 16
doubted d 0 9
doubting g 0 3
down b 0 2
downed d 0 2
downed n 0 3
doze b 0 0
dozed d 0 4
dozing g 0 3
draft b 0 1
drafted d 0 2
drafted n 0 3
drafting g 0 4
drag b 0 10
dragged d 0 8
dragged n 0 7
dragging g 0 15
drain b 0 7
drained d 0 3
drained n 0 4
dramatize b 0 3
dramatizes z 0 2
drape b 0 0
draped d 0 3
draped n 0 6
draw b 0 46
drawing g 0 27
drew d 1 63
drawn n 1 70
draws z 0 14
drawl b 0 0
drawled d 0 3
dream b 0 11
dreamed d 0 7
dreamed n 0 12
dreaming g 0 11
dreams z 0 2
dress b 0 14
dressed d 0 10
dressed n 0 26
dressing g 0 17
dry b 0 15
drying g 0 26
dried d 0 6
dried n 0 22
drift b 0 3
drifted d 0 5
drifted n 0 4
drifting g 0 11
drifts z 0 2
drill b 0 16
drilled n 0 5
drilling g 0 9
drink b 0 26
drinking g 0 42
drinks z 0 3
drank d 1 19
drip b 0 0
dripped d 0 5
dripping g 0 7
drive b 0 46
drives z 0 5
driving g 0 47

driven	n 1 44	edited	n 0 6	employed	n 0 43
drove	d 1 58	editing	g 0 3	employing	g 0 10
drop	b 0 34	educate	b 0 7	employs	z 0 9
dropped	d 0 76	educated	n 0 21	empty	b 0 0
dropped	n 0 25	educating	g 0 3	emptied	d 0 3
dropping	g 0 16	effect	b 0 16	emptied	n 0 5
drops	z 0 8	effected	n 0 11	empties	z 0 3
drown	b 0 3	effecting	g 0 3	emulate	b 0 3
drowned	d 0 4	effectuate	b 0 2	enable	b 0 23
drowned	n 0 2	ejaculate	b 0 0	enabled	d 0 10
drowning	g 0 4	ejaculated	d 0 3	enabled	n 0 2
drum	b 0 0	elaborate	b 0 6	enables	z 0 9
drummed	d 0 2	elaborated	n 0 3	enabling	g 0 13
drumming	g 0 4	elect	b 0 8	enact	b 0 7
duck	b 0 7	elected	d 0 2	enacted	d 0 2
ducked	d 0 5	elected	n 0 31	enacted	n 0 10
ducking	g 0 3	elicit	b 0 3	enacting	g 0 4
dump	b 0 3	elicited	d 0 3	enclose	b 0 0
dumped	d 0 7	elicited	n 0 3	enclosed	d 0 3
dumped	n 0 2	eliminate	b 0 26	enclosed	n 0 8
dumping	g 0 2	eliminated	d 0 6	encompass	b 0 4
duplicate	b 0 2	eliminated	n 0 16	encompassed	n 0 3
duplicated	n 0 2	eliminates	z 0 4	encounter	b 0 13
dwarf	b 0 2	eliminating	g 0 15	encountered	d 0 12
dwell	b 0 8	elude	b 0 0	encountered	n 0 18
dwelling	g 0 5	eluded	d 0 2	encounters	z 0 4
dwindle	b 0 2	eluding	g 0 2	encourage	b 0 46
dwindled	d 0 2	emancipate	b 0 2	encouraged	d 0 5
dwindling	g 0 4	emancipated	n 0 2	encouraged	n 0 24
dye	b 0 0	embark	b 0 5	encourages	z 0 5
dyed	n 0 4	embarrass	b 0 0	encouraging	g 0 15
dying	g 0 31	embarrassed	n 0 7	end	b 0 40
earn	b 0 16	embarrassing	g 0 11	ended	d 0 41
earned	d 0 9	embody	b 0 1	ended	n 0 18
earned	n 0 9	embodied	n 0 6	ending	g 0 27
earning	g 0 9	embodies	z 0 3	ends	z 0 13
earns	z 0 2	embodying	g 0 3	endear	b 0 0
ease	b 0 14	embrace	b 0 8	endeared	d 0 2
eased	d 0 2	embraced	d 0 4	endorse	b 0 6
eased	n 0 6	embraces	z 0 3	endorsed	n 0 3
easing	g 0 3	embracing	g 0 3	endow	b 0 2
eat	b 0 61	emerge	b 0 18	endowed	n 0 7
eating	g 0 30	emerged	d 0 23	endure	b 0 8
eats	z 0 2	emerged	n 0 3	endured	d 0 4
ate	d 1 16	emerges	z 0 9	endured	n 0 7
eaten	n 1 12	emerging	g 0 15	endures	z 0 2
echo	b 0 3	emit	b 0 1	enduring	g 0 10
echoed	d 0 7	emitted	d 0 2	enforce	b 0 9
echoing	g 0 2	emphasize	b 0 20	enforced	n 0 20
economize	b 0 3	emphasized	d 0 9	enforcing	g 0 5
economizing	g 0 2	emphasized	n 0 9	engage	b 0 14
edge	b 0 1	emphasizes	z 0 3	engaged	d 0 5
edged	d 0 7	emphasizing	g 0 4	engaged	n 0 42
edging	g 0 4	employ	b 0 9	engaging	g 0 4
edit	b 0 2	employed	d 0 6	engender	b 0 2

engendered	n 0 9	equating	g 0 2
engulf	b 0 0	eradicate	b 0 2
engulfed	d 0 2	erase	b 0 1
engulfed	n 0 3	erased	n 0 2
enhance	b 0 5	erasing	g 0 2
enhanced	n 0 5	erect	b 0 5
enjoy	b 0 44	erected	d 0 3
enjoyed	d 0 36	erected	n 0 15
enjoyed	n 0 21	erecting	g 0 3
enjoying	g 0 17	erode	b 0 0
enjoys	z 0 10	eroded	n 0 3
enlarge	b 0 7	erupt	b 0 2
enlarged	n 0 6	erupted	d 0 5
enlarging	g 0 2	erupted	n 0 2
enlist	b 0 5	escape	b 0 44
enlisted	d 0 5	escaped	d 0 8
enlisted	n 0 6	escaped	n 0 10
enrich	b 0 5	escapes	z 0 2
enriched	n 0 2	escaping	g 0 5
enroll	b 0 5	escort	b 0 4
enrolled	d 0 2	escorted	d 0 4
enrolled	n 0 7	escorting	g 0 2
enslave	b 0 2	establish	b 0 58
ensue	b 0 2	established	d 0 9
ensued	d 0 4	established	n 0 99
ensues	z 0 2	establishes	z 0 4
ensuing	g 0 4	establishing	g 0 25
ensure	b 0 8	esteem	b 0 0
ensuring	g 0 2	esteemed	d 0 2
entail	b 0 5	estimate	b 0 9
entails	z 0 8	estimated	d 0 8
enter	b 0 78	estimated	n 0 59
entered	d 0 76	estimating	g 0 2
entered	n 0 21	estrange	b 0 0
entering	g 0 24	estranged	n 0 2
enters	z 0 13	evaluate	b 0 13
entertain	b 0 14	evaluated	n 0 11
entertained	d 0 4	evaluating	g 0 7
entertained	n 0 7	even	b 0 28
entertaining	g 0 8	evoke	b 0 6
entitle	b 0 5	evoked	d 0 2
entitled	d 0 2	evoked	n 0 5
entitled	n 0 54	evokes	z 0 5
entitles	z 0 4	evolve	b 0 5
entreat	b 0 1	evolved	d 0 2
entreated	d 0 2	evolved	n 0 6
entrust	b 0 2	evolving	g 0 2
entrusted	n 0 2	exact	b 0 0
envy	b 0 3	exacting	g 0 2
envied	d 0 5	exacts	z 0 2
equal	b 0 6	exaggerate	b 0 8
equals	z 0 7	exaggerated	n 0 13
equate	b 0 8	exaggerating	g 0 4
equated	d 0 3	examine	b 0 33
equated	n 0 2	examined	d 0 11
examined	n 0 17		
examining	g 0 7		
exceed	b 0 18		
exceeded	d 0 2		
exceeded	n 0 3		
exceeding	g 0 6		
exceeds	z 0 10		
exchange	b 0 3		
exchanged	d 0 2		
exchanged	n 0 5		
exchanging	g 0 3		
excite	b 0 3		
excited	d 0 2		
excited	n 0 21		
exciting	g 0 2		
exclaim	b 0 1		
exclaimed	d 0 14		
exclaiming	g 0 4		
exclude	b 0 7		
excluded	n 0 8		
excludes	z 0 3		
excluding	g 0 3		
excuse	b 0 6		
excused	d 0 2		
execute	b 0 7		
executed	n 0 13		
exemplify	b 0 2		
exemplified	n 0 3		
exercise	b 0 23		
exercised	d 0 7		
exercised	n 0 11		
exercises	z 0 4		
exercising	g 0 5		
exert	b 0 11		
exerted	d 0 3		
exerted	n 0 10		
exerting	g 0 2		
exerts	z 0 3		
exhale	b 0 0		
exhaled	d 0 2		
exhaust	b 0 2		
exhausted	d 0 3		
exhausted	n 0 12		
exhausting	g 0 3		
exhibit	b 0 12		
exhibited	d 0 4		
exhibited	n 0 6		
exhibiting	g 0 6		
exhibits	z 0 3		
exist	b 0 59		
existed	d 0 27		
existed	n 0 13		
existing	g 0 60		
exists	z 0 42		
exonerate	b 0 2		

expand	b	0	13	extended	d	0	12	fastened	d	0	2
expanded	d	0	7	extended	n	0	43	fastened	n	0	12
expanded	n	0	13	extending	g	0	29	father	b	0	1
expanding	g	0	28	extends	z	0	12	fathered	d	0	2
expands	z	0	3	exterminate	b	0	2	fathom	b	0	3
expect	b	0	108	extract	b	0	5	favor	b	0	23
expected	d	0	30	extracted	d	0	2	favored	d	0	7
expected	n	0	157	extracted	n	0	7	favored	n	0	11
expecting	g	0	18	extracting	g	0	4	favoring	g	0	4
expects	z	0	22	extricate	b	0	2	favors	z	0	4
expel	b	0	2	exude	b	0	0	fear	b	0	30
expelled	n	0	4	exuded	d	0	2	feared	d	0	10
expend	b	0	0	eye	b	0	0	feared	n	0	4
expended	d	0	2	eyed	d	0	7	fearing	g	0	5
expended	n	0	10	eyeing	g	0	4	fears	z	0	4
experience	b	0	18	eying	g	0	2	feature	b	0	7
experienced	d	0	9	face	b	0	51	featured	d	0	3
experienced	n	0	43	faced	d	0	23	featured	n	0	5
experiences	z	0	5	faced	n	0	31	features	z	0	5
experiencing	g	0	7	faces	z	0	13	featuring	g	0	4
experiment	b	0	7	facing	g	0	34	feed	b	0	62
experimented	d	0	2	facilitate	b	0	5	feeding	g	0	22
experimented	n	0	4	facilitates	z	0	2	feeds	z	0	7
experimenting	g	0	7	fade	b	0	1	fed	d	1	8
expire	b	0	1	faded	d	0	8	fed	n	1	33
expired	d	0	4	faded	n	0	10	feel	b	0	201
explain	b	0	64	fading	g	0	5	feeling	g	0	41
explained	d	0	61	fail	b	0	37	feels	z	0	45
explained	n	0	19	failed	d	0	52	felt	d	1	302
explaining	g	0	13	failed	n	0	22	felt	n	1	54
explains	z	0	20	failing	g	0	17	fell	b	0	4
explode	b	0	6	fails	z	0	14	felling	g	0	2
exploded	d	0	4	fall	b	0	66	fetch	b	0	6
exploded	n	0	4	falling	g	0	32	field	b	0	2
exploding	g	0	7	falls	z	0	19	fielding	g	0	2
exploit	b	0	8	fell	d	1	87	fight	b	0	43
exploited	d	0	2	fallen	n	1	34	fighting	g	0	62
exploited	n	0	7	falsify	b	0	2	fights	z	0	3
explore	b	0	12	falter	b	0	2	fought	d	1	23
explored	d	0	2	faltered	d	0	3	fought	n	1	22
explored	n	0	9	fan	b	0	4	figure	b	0	20
exploring	g	0	5	fanned	d	0	4	figured	d	0	15
export	b	0	2	fanning	g	0	3	figured	n	0	6
exported	n	0	3	fans	z	0	2	figures	z	0	3
expose	b	0	7	fancy	b	0	3	figuring	g	0	5
exposed	d	0	4	fancied	d	0	2	file	b	0	33
exposed	n	0	30	farm	b	0	3	filed	d	0	12
exposes	z	0	2	farming	g	0	12	filed	n	0	21
exposing	g	0	4	fascinate	b	0	3	filing	g	0	19
express	b	0	27	fascinated	d	0	2	fill	b	0	50
expressed	d	0	24	fascinated	n	0	5	filled	d	0	31
expressed	n	0	51	fashion	b	0	3	filled	n	0	68
expresses	z	0	9	fashioned	d	0	4	filling	g	0	29
expressing	g	0	24	fashioned	n	0	3	fills	z	0	5
extend	b	0	31	fasten	b	0	4	filter	b	0	1

filtered	n 0 5	flaunted	d 0 2
filtering	g 0 4	flee	b 0 1
finance	b 0 18	fleeing	g 0 10
financed	n 0 16	fled	d 1 22
financing	g 0 21	fled	n 1 6
find	b 0 397	flex	b 0 1
finding	g 0 42	flexed	d 0 2
finds	z 0 59	flick	b 0 0
found	d 1 268	flicked	d 0 5
found	n 1 267	flicker	b 0 1
finger	b 0 0	flickered	d 0 2
fingered	d 0 3	fling	b 0 2
finish	b 0 24	flung	d 1 9
finished	d 0 31	flung	n 1 5
finished	n 0 56	flip	b 0 3
finishing	g 0 8	flipped	d 0 3
fire	b 0 10	flipping	g 0 2
fired	d 0 19	float	b 0 3
fired	n 0 25	floated	d 0 6
firing	g 0 23	floating	g 0 12
fit	b 0 39	flock	b 0 1
fits	z 0 10	flocked	d 0 2
fitting	g 0 10	flog	b 0 1
fitted	d 0 5	flogged	d 0 2
fitted	n 0 15	flood	b 0 2
fix	b 0 13	flooded	d 0 5
fixed	d 0 12	flooded	n 0 4
fixed	n 0 75	flooding	g 0 2
fixing	g 0 9	flop	b 0 0
flag	b 0 1	flopped	d 0 6
flags	z 0 2	flourish	b 0 4
flame	b 0 4	flourished	d 0 6
flaming	g 0 6	flourishes	z 0 2
flank	b 0 0	flow	b 0 13
flanked	d 0 2	flowed	d 0 4
flanked	n 0 3	flowed	n 0 2
flap	b 0 0	flowing	g 0 17
flapped	d 0 4	flows	z 0 4
flapping	g 0 4	flower	b 0 1
flare	b 0 0	flowered	n 0 2
flared	d 0 3	flowering	g 0 2
flared	n 0 2	flutter	b 0 0
flaring	g 0 3	fluttered	d 0 2
flash	b 0 6	fluttering	g 0 4
flashed	d 0 12	fly	b 0 18
flashed	n 0 4	flies	z 0 4
flashing	g 0 6	flying	g 0 35
flatten	b 0 1	flying	g 0 4
flattened	d 0 2	flew	d 1 27
flattened	n 0 4	flown	n 1 4
flattening	g 0 2	foam	b 0 1
flatter	b 0 1	foamed	n 0 8
flattered	d 0 2	foaming	g 0 2
flattered	n 0 5	focus	b 0 12
flaunt	b 0 0	focused	d 0 6
focused	n 0 6		
focuses	z 0 2		
focusing	g 0 5		
focussed	d 0 2		
foil	b 0 2		
fold	b 0 2		
folded	d 0 5		
folded	n 0 10		
folding	g 0 3		
follow	b 0 97		
followed	d 0 91		
followed	n 0 81		
following	g 0 192		
follows	z 0 77		
fool	b 0 5		
fooled	n 0 3		
fooling	g 0 2		
forbid	b 0 4		
forbidding	g 0 2		
forbids	z 0 5		
forbade	d 1 1		
forbidden	n 1 15		
force	b 0 24		
forced	d 0 19		
forced	n 0 62		
forces	z 0 6		
forcing	g 0 13		
forecast	b 0 2		
forecast	n 1 5		
forecasting	g 0 7		
forego	b 0 3		
foregoing	g 0 7		
foretell	b 0 1		
foresee	b 0 3		
foreseeing	g 0 2		
forestall	b 0 5		
forfeit	b 0 2		
forget	b 0 54		
forgetting	g 0 7		
forgot	d 1 17		
forgotten	n 1 38		
forgive	b 0 24		
forgave	d 1 2		
forgiven	n 1 6		
fork	b 0 1		
forked	n 0 4		
form	b 0 51		
formed	d 0 19		
formed	n 0 57		
forming	g 0 21		
forms	z 0 5		
formalize	b 0 2		
formalized	n 0 2		
formulate	b 0 9		
formulated	d 0 3		

formulated	n 0 8	furnished	n 0 19
formulating	g 0 4	furnishes	z 0 5
forsake	b 0 1	further	b 0 8
fort	b 0 3	furthered	d 0 2
fortify	b 0 2	furthering	g 0 2
fortified	d 0 2	fuse	b 0 2
fortified	n 0 5	fused	d 0 2
foster	b 0 3	fuss	b 0 2
fostered	n 0 6	fussing	g 0 2
fosters	z 0 3	gain	b 0 23
foul	b 0 1	gained	d 0 18
fouled	d 0 2	gained	n 0 21
found	b 0 1	gaining	g 0 14
founded	d 0 6	gang	b 0 2
founded	n 0 14	gape	b 0 0
founding	g 0 13	gaped	d 0 3
frame	b 0 4	gaping	g 0 2
framed	d 0 2	gasp	b 0 1
framed	n 0 12	gasped	d 0 5
framing	g 0 5	gasping	g 0 5
free	b 0 11	gather	b 0 20
freed	d 0 2	gathered	d 0 22
freed	n 0 10	gathered	n 0 10
freeing	g 0 3	gathering	g 0 13
frees	z 0 2	gaze	b 0 5
freeze	b 0 5	gazed	d 0 7
freezing	g 0 15	gazing	g 0 8
froze	d 1 4	generalize	b 0 5
frozen	n 1 27	generalized	n 0 9
frequent	b 0 2	generate	b 0 7
frighten	b 0 11	generated	d 0 2
frightened	d 0 2	generated	n 0 9
frightened	n 0 24	generates	z 0 5
frightening	g 0 14	generating	g 0 7
frown	b 0 1	germinate	b 0 2
frowned	d 0 7	gesture	b 0 0
frowning	g 0 12	gestured	d 0 3
frustrate	b 0 4	get	b 0 749
frustrated	n 0 9	gets	z 0 66
fry	b 0 2	getting	g 0 163
fried	n 0 6	got	d 1 338
fulfill	b 0 9	got	n 1 140
fulfilled	d 0 3	gotten	n 1 16
fulfilled	n 0 8	giggle	b 0 0
fulfilling	g 0 3	giggled	d 0 2
fulfills	z 0 2	give	b 0 387
fumble	b 0 0	gives	z 0 114
fumbled	d 0 5	giving	g 0 94
fumbling	g 0 4	gave	d 1 285
function	b 0 6	given	n 1 376
functioned	d 0 2	glance	b 0 10
functioning	g 0 5	glanced	d 0 25
functions	z 0 4	glancing	g 0 8
furnish	b 0 29	glare	b 0 1
furnished	d 0 4	glared	d 0 5
glaring	g 0 7
glaze	b 0 2
glazed	n 0 5
glazing	g 0 2
gleam	b 0 1
gleamed	d 0 4
gleaming	g 0 6
glide	b 0 2
glimpse	b 0 1
glimpsed	d 0 2
glimpsed	n 0 3
glint	b 0 0
glinted	d 0 2
glinting	g 0 5
glisten	b 0 2
glistened	d 0 4
glistening	g 0 6
gloat	b 0 0
gloated	d 0 2
glorify	b 0 2
glorified	n 0 4
glow	b 0 2
glowed	d 0 6
glowing	g 0 10
glower	b 0 0
glowered	d 0 2
glowering	g 0 3
glue	b 0 1
glued	d 0 2
glued	n 0 17
go	b 0 625
goes	z 0 89
going	g 0 396
gone	n 1 195
went	d 1 508
gobble	b 0 0
gobbled	d 0 2
gouge	b 0 1
gouged	d 0 2
gouging	g 0 3
govern	b 0 7
governed	n 0 14
governing	g 0 21
governs	z 0 2
grab	b 0 12
grabbed	d 0 19
grabbing	g 0 4
graduate	b 0 3
graduated	d 0 3
graduated	n 0 10
graduates	z 0 3
graduating	g 0 6
grant	b 0 14
granted	d 0 7
granted	n 0 49

granting	g	0	7	guide	b	0	18	hates	z	0	3
grasp	b	0	10	guided	d	0	4	hating	g	0	2
grasped	d	0	5	guided	n	0	16	haul	b	0	3
grasped	n	0	6	guides	z	0	3	hauled	d	0	3
grasping	g	0	2	guiding	g	0	10	hauled	n	0	6
greet	b	0	7	gulp	b	0	0	hauling	g	0	4
greeted	d	0	15	gulped	d	0	3	haunt	b	0	2
greeted	n	0	5	gush	b	0	0	haunted	d	0	2
grimace	b	0	2	gushed	d	0	5	haunted	n	0	6
grimaced	d	0	2	hail	b	0	6	haunting	g	0	2
grin	b	0	1	hailed	d	0	2	head	b	0	13
grinned	d	0	29	hailed	n	0	5	headed	d	0	23
grinning	g	0	7	halt	b	0	7	headed	n	0	36
grind	b	0	2	halted	d	0	10	heading	g	0	13
grinding	g	0	7	halted	n	0	2	heads	z	0	2
ground	d	1	4	halting	g	0	2	heal	b	0	2
ground	n	1	12	hammer	b	0	1	healed	d	0	3
grip	b	0	1	hammered	d	0	3	healed	n	0	3
gripped	d	0	9	hamper	b	0	2	healing	g	0	3
gripped	n	0	3	hampered	n	0	3	hear	b	0	153
gripping	g	0	6	hand	b	0	8	hearing	g	0	28
groan	b	0	0	handed	d	0	25	hears	z	0	7
groaned	d	0	3	handed	n	0	13	heard	d	1	129
grok	b	0	5	handing	g	0	6	heard	n	1	112
grokked	d	0	3	handle	b	0	34	heat	b	0	5
grope	b	0	1	handled	d	0	6	heated	n	0	16
groped	d	0	7	handled	n	0	20	heating	g	0	17
groping	g	0	4	handles	z	0	6	heave	b	0	1
ground	b	0	1	handling	g	0	15	heaved	d	0	4
grounded	d	0	2	hang	b	0	26	heaving	g	0	3
grounded	n	0	4	hanging	g	0	27	heed	b	0	7
group	b	0	4	hangs	z	0	4	help	b	0	214
grouped	n	0	5	hung	d	1	53	helped	d	0	40
grouping	g	0	4	hung	n	1	12	helped	n	0	26
grow	b	0	63	happen	b	0	63	helping	g	0	44
growing	g	0	107	happened	d	0	86	helps	z	0	28
grows	z	0	22	happened	n	0	63	herd	b	0	2
grew	d	1	65	happening	g	0	26	hesitate	b	0	10
grown	n	1	43	happens	z	0	40	hesitated	d	0	20
growl	b	0	0	harass	b	0	1	hibernate	b	0	2
growled	d	0	4	harassed	d	0	2	hide	b	0	18
grumble	b	0	5	harassed	n	0	4	hiding	g	0	16
grunt	b	0	1	harassing	g	0	2	hid	d	1	6
grunted	d	0	9	harbor	b	0	3	hidden	n	1	20
guarantee	b	0	2	harbored	n	0	2	hinder	b	0	0
guaranteed	d	0	2	hark	b	0	3	hindered	d	0	2
guaranteed	n	0	11	harvest	b	0	2	hint	b	0	1
guarantees	z	0	4	harvesting	g	0	2	hinted	n	0	6
guard	b	0	8	hasten	b	0	3	hints	z	0	2
guarded	n	0	4	hastened	d	0	6	hire	b	0	15
guarding	g	0	9	hastened	n	0	3	hired	d	0	6
guess	b	0	53	hastening	g	0	2	hired	n	0	19
guessed	d	0	7	hate	b	0	33	hiring	g	0	6
guessed	n	0	8	hated	d	0	18	hiss	b	0	0
guessing	g	0	8	hated	n	0	10	hissed	d	0	2

hissing	g	0	2	hurling	g	0	5	imposed	n	0	15
hit	b	0	38	hurry	b	0	18	imposes	z	0	4
hits	z	0	7	hurried	d	0	18	imposing	g	0	7
hitting	g	0	17	hurried	n	0	5	impress	b	0	3
hit	d	1	38	hurrying	g	0	4	impressed	d	0	7
hit	n	1	26	hurt	b	0	15	impressed	n	0	23
hitch	b	0	0	hurting	g	0	3	impressing	g	0	2
hitched	n	0	3	hurts	z	0	3	improve	b	0	39
hitching	g	0	3	hurt	n	1	9	improved	d	0	7
hold	b	0	144	hustle	b	0	2	improved	n	0	48
holding	g	0	61	identify	b	0	26	improves	z	0	11
holds	z	0	39	identified	d	0	11	improving	g	0	16
held	d	1	126	identified	n	0	35	improvise	b	0	2
held	n	1	138	identifies	z	0	6	improvised	d	0	2
holler	b	0	0	identifying	g	0	3	inactivate	b	0	2
hollered	d	0	2	ignite	b	0	2	incite	b	0	3
hollering	g	0	3	ignore	b	0	19	incited	d	0	2
honor	b	0	14	ignored	d	0	13	include	b	0	113
honored	d	0	2	ignored	n	0	16	included	d	0	41
honored	n	0	22	ignores	z	0	5	included	n	0	56
honoring	g	0	8	ignoring	g	0	4	includes	z	0	45
honors	z	0	2	illumine	b	0	1	including	g	0	5
hook	b	0	1	illumined	d	0	2	incorporate	b	0	2
hooked	d	0	2	illustrate	b	0	17	incorporated	d	0	2
hooked	n	0	5	illustrated	n	0	36	incorporated	n	0	11
hope	b	0	68	illustrates	z	0	7	incorporates	z	0	3
hoped	d	0	33	illustrating	g	0	4	increase	b	0	82
hoped	n	0	15	imagine	b	0	61	increased	d	0	38
hopes	z	0	18	imagined	d	0	12	increased	n	0	108
hoping	g	0	30	imagined	n	0	15	increases	z	0	30
hop	b	0	1	imagines	z	0	3	increasing	g	0	74
hopped	d	0	5	imitate	b	0	5	incur	b	0	5
hopping	g	0	4	imitated	d	0	2	incurred	n	0	8
house	b	0	9	imitated	n	0	2	indicate	b	0	80
housed	d	0	4	imitates	z	0	2	indicated	d	0	59
housed	n	0	8	imitating	g	0	2	indicated	n	0	49
houses	z	0	3	impair	b	0	4	indicates	z	0	40
housing	g	0	29	impaired	n	0	7	indicating	g	0	16
hover	b	0	4	impaled	n	0	2	induce	b	0	9
huddle	b	0	0	impart	b	0	4	induced	n	0	12
huddled	d	0	6	imparted	n	0	3	induces	z	0	3
huddled	n	0	4	impinge	b	0	3	inducing	g	0	4
huddling	g	0	3	impinging	g	0	5	indulge	b	0	9
hug	b	0	2	implement	b	0	3	indulged	d	0	5
hugged	d	0	2	implementing	g	0	3	infer	b	0	1
hugging	g	0	7	imply	b	0	13	inferred	d	0	2
hum	b	0	1	implied	d	0	5	inflict	b	0	4
hummed	d	0	2	implied	n	0	12	inflicted	d	0	3
humming	g	0	2	implies	z	0	16	inflicting	g	0	3
hunt	b	0	5	implying	g	0	7	influence	b	0	19
hunted	d	0	2	import	b	0	3	influenced	n	0	15
hunted	n	0	5	imported	d	0	2	influences	z	0	3
hunting	g	0	31	imported	n	0	6	influencing	g	0	2
hurl	b	0	3	impose	b	0	9	inform	b	0	7
hurled	d	0	3	imposed	d	0	4	informed	d	0	22

informed	n 0 35	insulate	b 0 2
informing	g 0 4	insulated	n 0 4
informs	z 0 6	insulating	g 0 2
infuriate	b 0 1	insult	b 0 2
infuriated	d 0 3	insure	b 0 24
ingest	b 0 0	insured	n 0 5
ingested	d 0 2	insuring	g 0 6
ingested	n 0 2	integrate	b 0 7
inherit	b 0 4	integrated	n 0 11
inherited	d 0 6	integrates	z 0 2
inherited	n 0 10	integrating	g 0 2
inhibit	b 0 8	intend	b 0 15
inhibited	d 0 2	intended	d 0 10
inhibited	n 0 3	intended	n 0 35
inhibiting	g 0 2	intends	z 0 6
inhibits	z 0 2	intensify	b 0 4
initiate	b 0 5	intensified	n 0 3
initiated	d 0 4	intensifying	g 0 2
initiated	n 0 8	interact	b 0 2
initiating	g 0 4	intercept	b 0 3
inject	b 0 6	intercepted	d 0 3
injecting	g 0 5	interest	b 0 3
injure	b 0 0	interested	d 0 3
injured	d 0 2	interested	n 0 98
injured	n 0 18	interests	z 0 2
inquire	b 0 6	interfere	b 0 9
inquired	d 0 14	interfered	d 0 4
inquired	n 0 2	interferes	z 0 2
inquiring	g 0 5	interfering	g 0 6
insert	b 0 10	interpenetrate	b 0 1
inserted	d 0 5	interpenetrates	z 0 2
inserted	n 0 11	interpret	b 0 11
insist	b 0 27	interpreted	d 0 3
insisted	d 0 39	interpreted	n 0 21
insisted	n 0 4	interpreting	g 0 2
insisting	g 0 6	interprets	z 0 3
insists	z 0 10	interrupt	b 0 4
inspect	b 0 12	interrupted	d 0 10
inspecting	g 0 2	interrupted	n 0 8
inspire	b 0 3	intersect	b 0 6
inspired	d 0 8	intersecting	g 0 2
inspired	n 0 17	intertwined	n 0 3
inspiring	g 0 4	intervene	b 0 2
install	b 0 8	intervened	d 0 4
installed	d 0 5	interview	b 0 2
installed	n 0 30	interviewed	d 0 7
installing	g 0 5	interviewed	n 0 5
institute	b 0 1	interviewing	g 0 5
instituted	d 0 3	intimate	b 0 0
instituted	n 0 9	intimated	d 0 4
instruct	b 0 3	intimidate	b 0 2
instructed	d 0 2	intimidated	n 0 3
instructed	n 0 14	intone	b 0 0
instructing	g 0 2	intoned	d 0 4
instructs	z 0 2	intrigued	n 0 2
introduce	b 0 11		
introduced	d 0 15		
introduced	n 0 37		
introduces	z 0 4		
introducing	g 0 9		
inure	b 0 2		
inured	n 0 2		
invade	b 0 5		
invaded	n 0 5		
invading	g 0 3		
invalidate	b 0 2		
invent	b 0 7		
invented	d 0 5		
invented	n 0 8		
invest	b 0 3		
invested	n 0 11		
investigate	b 0 11		
investigated	d 0 2		
investigated	n 0 16		
investigating	g 0 8		
invite	b 0 10		
invited	d 0 11		
invited	n 0 15		
invites	z 0 7		
inviting	g 0 6		
invoke	b 0 4		
invoked	n 0 5		
invoking	g 0 4		
involve	b 0 31		
involved	d 0 23		
involved	n 0 124		
involves	z 0 41		
involving	g 0 30		
iodinate	b 0 1		
iodinated	n 0 7		
iodinating	g 0 3		
ionize	b 0 0		
ionized	n 0 3		
ionizing	g 0 6		
iron	b 0 1		
ironed	n 0 2		
ironing	g 0 5		
isolate	b 0 8		
isolated	n 0 35		
isolating	g 0 5		
issue	b 0 15		
issued	d 0 20		
issued	n 0 30		
issues	z 0 3		
issuing	g 0 4		
itch	b 0 3		
itching	g 0 4		
itemize	b 0 0		
itemized	n 0 3		
itemizing	g 0 2		

jab	b 0 0	knitted	n 0 7
jabbed	d 0 2	knitted	d 0 1
jabbing	g 0 2	knit	n 1 6
jam	b 0 2	knock	b 0 11
jammed	n 0 7	knocked	d 0 17
jeopardize	b 0 4	knocked	n 0 14
jerk	b 0 1	knocking	g 0 4
jerked	d 0 12	know	b 0 680
jerking	g 0 3	knowing	g 0 49
jingle	b 0 0	knows	z 0 99
jingled	d 0 2	knew	d 1 395
join	b 0 65	known	n 1 245
joined	d 0 33	label	b 0 1
joined	n 0 23	labeled	d 0 2
joining	g 0 15	labeled	n 0 7
joins	z 0 2	labeling	g 0 4
joke	b 0 3	labor	b 0 4
joking	g 0 5	labored	d 0 3
jolt	b 0 2	labored	n 0 2
journey	b 0 1	lack	b 0 13
journeyed	d 0 2	lacked	d 0 15
judge	b 0 16	lacked	n 0 4
judged	d 0 3	lacking	g 0 32
judged	n 0 12	lacks	z 0 6
judging	g 0 9	lag	b 0 2
jump	b 0 15	land	b 0 9
jumped	d 0 32	landed	d 0 12
jumped	n 0 3	landed	n 0 3
jumping	g 0 7	landing	g 0 13
justify	b 0 26	lapse	b 0 2
justified	n 0 22	lapsed	d 0 2
justifying	g 0 3	lash	b 0 4
keep	b 0 261	lashed	d 0 3
keeping	g 0 57	lashing	g 0 2
keeps	z 0 19	last	b 0 23
kept	d 1 116	lasted	d 0 11
kept	n 1 70	lasting	g 0 12
keynote	b 0 3	laugh	b 0 9
kick	b 0 4	laughed	d 0 46
kicked	d 0 10	laughed	n 0 5
kicked	n 0 8	laughing	g 0 28
kicking	g 0 11	launch	b 0 7
kill	b 0 60	launched	d 0 3
killed	d 0 34	launched	n 0 17
killed	n 0 41	launches	z 0 2
killing	g 0 11	launching	g 0 2
kills	z 0 6	lay	b 0 48
kiss	b 0 9	laying	g 0 9
kissed	d 0 15	lays	z 0 5
kissing	g 0 6	laid	d 1 24
kneel	b 0 5	laid	n 1 53
kneeling	g 0 5	lead	b 0 82
knelt	d 1 7	leading	g 0 68
knelt	n 1 1	leads	z 0 31
knit	b 0 2	led	d 1 83
led	n 1 49		
leak	b 0 0		
leaked	d 0 4		
lean	b 0 7		
leaned	d 0 37		
leaning	g 0 15		
leap	b 0 8		
leaped	d 0 18		
leaped	n 0 2		
leaping	g 0 2		
learn	b 0 84		
learned	d 0 54		
learned	n 0 58		
learning	g 0 48		
learns	z 0 10		
lease	b 0 3		
leased	n 0 2		
leasing	g 0 3		
leave	b 0 195		
leaves	z 0 26		
leaving	g 0 88		
left	d 1 157		
left	n 1 181		
lecture	b 0 2		
lecturing	g 0 3		
leer	b 0 0		
leered	d 0 3		
leering	g 0 4		
lend	b 0 14		
lending	g 0 6		
lends	z 0 4		
lent	d 1 3		
lent	n 1 2		
lengthen	b 0 2		
lengthened	n 0 2		
lengthening	g 0 3		
lessen	b 0 5		
lessened	d 0 3		
lessened	n 0 6		
lessening	g 0 2		
let	b 0 335		
lets	z 0 5		
letting	g 0 30		
let	d 1 37		
let	n 1 4		
level	b 0 2		
leveled	d 0 4		
leveled	n 0 9		
leveling	g 0 9		
levy	b 0 3		
liberate	b 0 4		
liberated	n 0 7		
license	b 0 1		
licensed	n 0 6		
licensing	g 0 5		

lick	b 0 3		
licked	d 0 7		
licked	n 0 3		
lie	b 0 53		
lying	g 0 36		
lied	d 0 6		
lies	z 0 41		
lift	b 0 18		
lifted	d 0 34		
lifted	n 0 9		
lifting	g 0 7		
light	b 0 10		
lighted	n 0 23		
lighting	g 0 16		
lit	b 0 9		
lit	d 1 9		
lit	n 1 7		
lighten	b 0 0		
lightened	d 0 2		
lightened	n 0 2		
like	b 0 211		
liked	d 0 45		
liked	n 0 13		
likes	z 0 18		
liking	g 0 4		
limit	b 0 17		
limited	d 0 6		
limited	n 0 100		
limiting	g 0 11		
limits	z 0 4		
line	b 0 4		
lined	d 0 7		
lined	n 0 9		
lining	g 0 2		
linger	b 0 7		
lingered	d 0 2		
lingering	g 0 5		
lingers	z 0 2		
link	b 0 4		
linked	n 0 15		
linking	g 0 5		
list	b 0 7		
listed	d 0 11		
listed	n 0 33		
listing	g 0 2		
lists	z 0 6		
listen	b 0 51		
listened	d 0 29		
listening	g 0 39		
listens	z 0 2		
live	b 0 157		
lived	d 0 72		
lived	n 0 43		
lives	z 0 30		
living	g 0 170		
load	b 0 3		
loaded	n 0 21		
loading	g 0 5		
loathe	b 0 0		
loathed	d 0 4		
locate	b 0 16		
located	d 0 4		
located	n 0 60		
locating	g 0 11		
lock	b 0 2		
locked	d 0 9		
locked	n 0 21		
locking	g 0 31		
lodge	b 0 2		
lodging	g 0 4		
log	b 0 0		
logged	n 0 2		
logging	g 0 4		
long	b 0 3		
longed	d 0 4		
longed	n 0 3		
longing	g 0 5		
look	b 0 303		
looked	d 0 329		
looked	n 0 38		
looking	g 0 167		
looks	z 0 69		
loom	b 0 1		
loomed	d 0 3		
looming	g 0 10		
loose	b 0 3		
loosen	b 0 3		
loosened	n 0 3		
loot	b 0 0		
looted	n 0 2		
looting	g 0 3		
lose	b 0 58		
loses	z 0 15		
losing	g 0 28		
lost	d 1 49		
lost	n 1 124		
lounge	b 0 1		
lounged	d 0 3		
lounging	g 0 4		
love	b 0 56		
loved	d 0 45		
loved	n 0 11		
loves	z 0 17		
loving	g 0 14		
lower	b 0 7		
lowered	d 0 10		
lowered	n 0 11		
lowering	g 0 4		
lug	b 0 1		
lugged	d 0 3		
lugged	n 0 2		
lunge	b 0 1		
lunged	d 0 4		
lurch	b 0 0		
lurched	d 0 5		
lurching	g 0 2		
lure	b 0 3		
lured	d 0 3		
lurk	b 0 1		
lurked	d 0 3		
lurking	g 0 3		
magnify	b 0 0		
magnified	n 0 5		
magnifying	g 0 3		
mail	b 0 5		
mailed	d 0 3		
mailed	n 0 13		
mailing	g 0 4		
maintain	b 0 60		
maintained	d 0 13		
maintained	n 0 35		
maintaining	g 0 28		
maintains	z 0 16		
make	b 0 791		
makes	z 0 168		
making	g 0 231		
made	d 1 466		
made	n 1 656		
man	b 0 3		
manned	n 0 12		
manning	g 0 2		
manage	b 0 20		
managed	d 0 23		
managed	n 0 13		
manages	z 0 4		
managing	g 0 8		
maneuver	b 0 2		
maneuvered	d 0 3		
maneuvering	g 0 3		
manifest	b 0 4		
manifested	d 0 3		
manifested	n 0 3		
manipulate	b 0 6		
manipulating	g 0 2		
manufacture	b 0 2		
manufactured	d 0 3		
manufactured	n 0 8		
manufactures	z 0 2		
manufacturing	g 0 22		
mar	b 0 2		
marred	n 0 4		
mars	z 0 2		
march	b 0 10		
marched	d 0 6		
marched	n 0 3		

marches	z	0	3	melted	n	0	7	missing	g	0	32
marching	g	0	15	melting	g	0	19	mistrust	b	0	0
mark	b	0	18	memorize	b	0	3	mistrusted	d	0	2
marked	d	0	15	memorized	n	0	2	misuse	b	0	2
marked	n	0	69	menace	b	0	0	mitigate	b	0	1
marking	g	0	10	menaced	n	0	2	mitigates	z	0	2
marks	z	0	14	menacing	g	0	4	mitigating	g	0	2
market	b	0	0	mend	b	0	2	mix	b	0	11
marketed	n	0	3	mending	g	0	3	mixed	n	0	36
marketing	g	0	37	mention	b	0	33	mixing	g	0	8
marry	b	0	18	mentioned	d	0	18	moan	b	0	1
married	d	0	22	mentioned	n	0	61	moaned	d	0	2
married	n	0	82	mentioning	g	0	7	mobilize	b	0	2
marries	z	0	3	mentions	z	0	6	mobilized	n	0	4
marrying	g	0	3	merge	b	0	10	mobilizing	g	0	3
marshal	b	0	2	merged	n	0	3	mock	b	0	3
marvel	b	0	3	merges	z	0	2	mocked	n	0	2
masquerade	b	0	1	merging	g	0	4	mocking	g	0	5
masquerades	z	0	2	merit	b	0	2	modernize	b	0	1
mass	b	0	3	merited	d	0	4	modernized	n	0	2
master	b	0	7	merits	z	0	3	modernizing	g	0	3
mastered	n	0	4	mesh	b	0	2	modify	b	0	6
match	b	0	26	mess	b	0	2	modified	n	0	13
matched	d	0	2	messing	g	0	2	modifies	z	0	2
matched	n	0	14	meter	b	0	0	modifying	g	0	4
matches	z	0	4	metered	n	0	4	moisten	b	0	2
matching	g	0	31	metering	g	0	2	mold	b	0	2
mate	b	0	4	milk	b	0	0	molded	n	0	11
mated	n	0	4	milks	z	0	2	molding	g	0	7
mating	g	0	8	mind	b	0	38	mollify	b	0	2
materialize	b	0	3	minded	n	0	2	monopolize	b	0	4
matriculate	b	0	2	mingle	b	0	2	mop	b	0	2
matriculated	n	0	2	mingled	d	0	3	mopped	d	0	2
matter	b	0	25	mingled	n	0	5	mopped	n	0	2
mattered	d	0	4	minimize	b	0	16	mopping	g	0	3
matters	z	0	5	minimized	d	0	2	motivate	b	0	1
mature	b	0	7	minimized	n	0	3	motivated	n	0	8
maturing	g	0	3	minimizing	g	0	3	motivates	z	0	3
mean	b	0	158	minister	b	0	0	motivating	g	0	3
meaning	g	0	12	ministered	d	0	2	mount	b	0	4
means	z	0	103	ministering	g	0	3	mounted	d	0	13
meant	d	1	70	mirror	b	0	1	mounted	n	0	32
meant	n	1	30	mirrors	z	0	3	mounting	g	0	10
measure	b	0	30	misinterpret	b	0	2	mounts	z	0	3
measured	d	0	7	misinterpreted	n	0	2	mourn	b	0	2
measured	n	0	59	mislead	b	0	0	mourned	d	0	2
measures	z	0	3	misleading	g	0	8	mourning	g	0	8
measuring	g	0	29	misled	n	1	2	move	b	0	135
meet	b	0	142	misled	d	1	1	moved	d	0	138
meeting	g	0	34	misrepresent	b	0	0	moved	n	0	43
meets	z	0	33	misrepresents	z	0	2	moves	z	0	27
met	d	1	80	miss	b	0	20	moving	g	0	104
met	n	1	49	missed	d	0	17	multiply	b	0	10
melt	b	0	4	missed	n	0	23	multiplied	d	0	2
melted	d	0	2	misses	z	0	3	multiplied	n	0	5

multiplies	z	0	2	nodded	d	0	49	occupy	b	0	16
multiplying	g	0	8	nodded	n	0	2	occupied	d	0	9
mumble	b	0	0	nodding	g	0	7	occupied	n	0	27
mumbled	d	0	5	nominate	b	0	3	occupies	z	0	4
murder	b	0	4	nominated	n	0	7	occupying	g	0	7
murdered	n	0	9	note	b	0	54	occur	b	0	43
murdering	g	0	3	noted	d	0	27	occurred	d	0	47
murmur	b	0	1	noted	n	0	63	occurred	n	0	20
murmured	d	0	17	noting	g	0	17	occurring	g	0	21
murmuring	g	0	4	notes	z	0	4	occurs	z	0	27
muse	b	0	0	notice	b	0	29	offend	b	0	4
mused	d	0	4	noticed	d	0	28	offended	n	0	2
muster	b	0	2	noticed	n	0	22	offer	b	0	68
mutter	b	0	1	noticing	g	0	5	offered	d	0	43
muttered	d	0	16	notify	b	0	8	offered	n	0	40
muttering	g	0	7	notified	d	0	2	offering	g	0	22
nail	b	0	0	notified	n	0	2	offers	z	0	44
nailed	d	0	3	nudge	b	0	1	officiate	b	0	1
nailed	n	0	8	nudged	d	0	2	officiated	d	0	3
name	b	0	21	number	b	0	2	offset	b	0	7
named	d	0	13	numbered	d	0	3	offset	n	1	2
named	n	0	71	numbered	n	0	6	omit	b	0	1
naming	g	0	3	numbering	g	0	6	omits	z	0	2
narrow	b	0	0	nurture	b	0	2	omitted	n	0	13
narrowed	d	0	3	obey	b	0	8	omitting	g	0	6
narrowed	n	0	6	obeyed	d	0	5	ooze	b	0	1
narrowing	g	0	3	obeyed	n	0	2	oozed	d	0	2
narrows	z	0	3	obeying	g	0	3	open	b	0	56
near	b	0	4	object	b	0	12	opened	d	0	94
neared	d	0	3	objected	d	0	12	opened	n	0	37
nearing	g	0	12	objects	z	0	3	opening	g	0	56
necessitate	b	0	5	oblige	b	0	0	opens	z	0	16
necessitated	d	0	4	obliged	d	0	3	operate	b	0	48
necessitated	n	0	7	obliged	n	0	18	operated	d	0	8
necessitates	z	0	3	obliterate	b	0	2	operated	n	0	19
neck	b	0	2	obscure	b	0	6	operates	z	0	15
need	b	0	162	obscured	d	0	2	operating	g	0	87
needed	d	0	57	obscured	n	0	5	oppose	b	0	15
needed	n	0	130	observe	b	0	25	opposed	d	0	9
needing	g	0	5	observed	d	0	15	opposed	n	0	32
needs	z	0	59	observed	n	0	59	opposes	z	0	2
negate	b	0	2	observes	z	0	8	opposing	g	0	13
neglect	b	0	4	observing	g	0	13	opt	b	0	0
neglected	d	0	2	obsess	b	0	0	opted	d	0	2
neglected	n	0	16	obsessed	n	0	5	ordain	b	0	2
neglecting	g	0	5	obsesses	z	0	2	ordained	n	0	4
negotiate	b	0	10	obstruct	b	0	4	order	b	0	17
negotiated	d	0	2	obstructed	d	0	2	ordered	d	0	28
negotiated	n	0	5	obstructed	n	0	2	ordered	n	0	41
negotiating	g	0	8	obtain	b	0	42	ordering	g	0	2
nest	b	0	1	obtained	d	0	8	organize	b	0	14
nested	d	0	2	obtained	n	0	107	organized	d	0	4
nested	n	0	2	obtaining	g	0	9	organized	n	0	52
nesting	g	0	2	obtrude	b	0	0	organizing	g	0	8
nod	b	0	4	obtrudes	z	0	2	orient	b	0	0

oriented	n 0 11	owed	n 0 3
orienting	g 0 3	owes	z 0 5
originate	b 0 6	owing	g 0 4
originated	d 0 10	own	b 0 22
originated	n 0 5	owned	d 0 15
originates	z 0 2	owned	n 0 19
originating	g 0 3	owns	z 0 13
oust	b 0 3	pace	b 0 4
outdistance	b 0 0	paced	d 0 10
outdistanced	d 0 2	pacing	g 0 4
outdo	b 0 3	pacify	b 0 2
outface	b 0 2	pack	b 0 11
outgrow	b 0 4	packed	d 0 7
outlaw	b 0 0	packed	n 0 12
outlawed	n 0 4	packing	g 0 14
outline	b 0 4	package	b 0 1
outlined	n 0 5	packaged	n 0 6
outlining	g 0 2	packaging	g 0 7
outnumber	b 0 2	pad	b 0 1
outrage	b 0 1	padded	n 0 5
outraged	n 0 5	padding	g 0 2
outrun	b 0 3	paint	b 0 19
outweigh	b 0 2	painted	d 0 9
outweighed	d 0 2	painted	n 0 31
outweighed	n 0 2	painting	g 0 32
overcome	b 0 18	paints	z 0 4
overcomes	z 0 7	panic	b 0 2
overcoming	g 0 6	parallel	b 0 2
overcame	d 1 3	paralleled	n 0 3
overcome	n 1 8	paralyze	b 0 1
overflow	b 0 0	paralyzed	n 0 2
overflowed	d 0 2	paralyzes	z 0 2
overflowing	g 0 2	pardon	b 0 4
overhear	b 0 0	pardoned	n 0 2
overheard	d 1 2	pare	b 0 2
overheard	n 1 4	park	b 0 1
overlap	b 0 4	parked	d 0 8
overlapped	n 0 2	parked	n 0 25
overlapping	g 0 3	parking	g 0 27
overload	b 0 2	parody	b 0 0
overlook	b 0 4	parodied	d 0 2
overlooked	d 0 2	part	b 0 3
overlooked	n 0 5	parted	d 0 2
overlooking	g 0 2	parted	n 0 3
overlooks	z 0 4	parting	g 0 3
overreach	b 0 2	partake	b 0 1
overreached	d 0 2	partakes	z 0 2
overshadow	b 0 2	participate	b 0 22
overtake	b 0 3	participated	d 0 5
overtaken	n 1 1	participated	n 0 8
overtook	d 1 1	participates	z 0 7
overthrow	b 0 3	participating	g 0 15
overthrown	n 1 3	pass	b 0 66
owe	b 0 10	passed	d 0 91
owed	d 0 12	passed	n 0 66
passes	z 0 16		
passing	g 0 59		
patrol	b 0 2		
patrolling	g 0 3		
patronize	b 0 1		
patronized	n 0 2		
patronizing	g 0 2		
pat	b 0 1		
patted	d 0 7		
patting	g 0 4		
pause	b 0 5		
paused	d 0 25		
paused	n 0 3		
pausing	g 0 6		
pave	b 0 2		
paved	n 0 4		
paving	g 0 2		
pay	b 0 133		
paying	g 0 26		
pays	z 0 17		
paid	d 0 50		
paid	n 0 95		
peck	b 0 1		
pecked	d 0 2		
peel	b 0 2		
peeled	d 0 2		
peeled	n 0 3		
peeling	g 0 6		
peer	b 0 2		
peered	d 0 19		
peering	g 0 9		
penetrate	b 0 7		
penetrated	d 0 5		
penetrated	n 0 3		
penetrating	g 0 2		
people	b 0 2		
perceive	b 0 13		
perceived	d 0 3		
perceived	n 0 9		
perceives	z 0 3		
perfect	b 0 0		
perfected	n 0 5		
perfecting	g 0 3		
perform	b 0 29		
performed	d 0 11		
performed	n 0 24		
performing	g 0 17		
performs	z 0 4		
perish	b 0 2		
permeate	b 0 1		
permeated	n 0 3		
permeates	z 0 2		
permit	b 0 66		
permits	z 0 25		
permitted	d 0 14		

permitted	n	0	43
permitting	g	0	9
perpetuate	b	0	5
perpetuating	g	0	4
persist	b	0	6
persisted	d	0	8
persisted	n	0	2
persisting	g	0	2
persists	z	0	7
personify	b	0	0
personifies	z	0	3
persuade	b	0	17
persuaded	d	0	6
persuaded	n	0	15
persuading	g	0	4
pertain	b	0	2
pertaining	g	0	5
pertains	z	0	5
pervade	b	0	0
pervades	z	0	2
pervading	g	0	2
petition	b	0	1
petitioned	d	0	4
pet	b	0	0
petted	n	0	2
petting	g	0	2
phone	b	0	5
phoned	d	0	3
phoned	n	0	2
phones	z	0	2
photograph	b	0	3
photographed	n	0	4
photographing	g	0	3
photographs	z	0	2
phrase	b	0	1
phrased	n	0	2
phrasing	g	0	2
pick	b	0	49
picked	d	0	51
picked	n	0	27
picking	g	0	13
picks	z	0	3
picture	b	0	2
pictured	d	0	2
pictured	n	0	2
picturing	g	0	2
pierce	b	0	0
pierced	n	0	3
piercing	g	0	3
pile	b	0	3
piled	d	0	7
piled	n	0	9
piling	g	0	6
pillage	b	0	2
pilot	b	0	2
pin	b	0	2
pinned	n	0	3
pinch	b	0	2
pinched	d	0	2
pinched	n	0	5
pinching	g	0	2
pinpoint	b	0	3
pioneer	b	0	1
pioneered	n	0	2
pioneering	g	0	3
pitch	b	0	1
pitched	d	0	4
pitched	n	0	4
pitching	g	0	11
pity	b	0	1
pitied	d	0	2
place	b	0	74
placed	d	0	25
placed	n	0	101
places	z	0	8
placing	g	0	25
plague	b	0	3
plagued	d	0	3
plagued	n	0	2
plan	b	0	30
planned	d	0	19
planned	n	0	56
planning	g	0	11
planning	g	0	73
plans	z	0	11
plant	b	0	4
planted	d	0	5
planted	n	0	6
planting	g	0	3
play	b	0	110
played	d	0	66
played	n	0	38
playing	g	0	84
plays	z	0	34
plead	b	0	5
pleaded	d	0	7
pleading	g	0	11
please	b	0	47
pleased	d	0	11
pleased	n	0	30
pleases	z	0	2
pledge	b	0	0
pledged	d	0	2
pledged	n	0	3
ply	b	0	0
plied	d	0	2
plot	b	0	4
plotted	d	0	2
plotted	n	0	8
plotting	g	0	2
plow	b	0	3
plowed	n	0	4
plowing	g	0	8
pluck	b	0	1
plucked	d	0	4
plug	b	0	2
plugged	n	0	2
plump	b	0	0
plumped	d	0	2
plunge	b	0	1
plunged	d	0	10
plunged	n	0	5
plunging	g	0	3
point	b	0	26
pointed	d	0	48
pointed	n	0	24
pointing	g	0	26
points	z	0	19
poise	b	0	0
poised	d	0	2
poised	n	0	10
poison	b	0	1
poisoned	d	0	2
poisoned	n	0	2
poisoning	g	0	2
poke	b	0	1
poked	d	0	3
pokes	z	0	3
poking	g	0	5
polish	b	0	4
polished	d	0	3
polished	n	0	11
polishing	g	0	2
ponder	b	0	1
pondered	d	0	3
pondering	g	0	2
pool	b	0	3
pooling	g	0	2
pop	b	0	2
popped	d	0	6
popping	g	0	6
pops	z	0	2
portray	b	0	6
portrayed	n	0	6
portraying	g	0	2
portrays	z	0	5
pose	b	0	9
posed	d	0	3
posed	n	0	4
posing	g	0	3
position	b	0	4
possess	b	0	17
possessed	d	0	18
possessed	n	0	5
possesses	z	0	8

possessing g 0 6
post b 0 2
posted d 0 3
posted n 0 8
postpone b 0 7
postponed n 0 8
postponing g 0 3
postulate b 0 3
postulated n 0 7
pound b 0 2
pounded d 0 4
pounding g 0 5
pour b 0 7
poured d 0 21
poured n 0 8
pouring g 0 9
pours z 0 2
power b 0 2
powered n 0 2
practice b 0 14
practiced d 0 2
practiced n 0 6
practicing g 0 14
praise b 0 4
praised d 0 8
praised n 0 5
praises z 0 2
praising g 0 2
pray b 0 12
prayed d 0 8
prayed n 0 4
praying g 0 3
preach b 0 8
preached d 0 6
preached n 0 2
preaching g 0 9
precede b 0 3
preceded d 0 9
preceded n 0 6
preceding g 0 29
preceeding g 0 2
precipitate b 0 0
precipitated d 0 3
precipitated n 0 6
precipitating g 0 2
preclude b 0 4
predict b 0 8
predicted d 0 5
predicted n 0 13
predicting g 0 6
predicts z 0 3
prefer b 0 27
preferred d 0 16
preferred n 0 10
prefers z 0 5
prejudice b 0 2
prejudiced n 0 4
prepare b 0 35
prepared d 0 12
prepared n 0 90
prepares z 0 4
preparing g 0 22
prescribe b 0 5
prescribed d 0 2
prescribed n 0 12
present b 0 39
presented d 0 16
presented n 0 66
presenting g 0 10
presents z 0 27
preserve b 0 31
preserved d 0 2
preserved n 0 17
preserves z 0 3
preserving g 0 10
preside b 0 2
presiding g 0 10
press b 0 27
pressed d 0 12
pressed n 0 17
presses z 0 2
pressing g 0 24
presume b 0 3
presumed d 0 2
presumed n 0 10
presumes z 0 3
presuming g 0 2
presuppose b 0 1
presupposes z 0 2
pretend b 0 7
pretended d 0 3
pretended n 0 3
pretending g 0 12
pretends z 0 2
prevail b 0 7
prevailed d 0 7
prevailing g 0 17
prevails z 0 7
prevent b 0 83
prevented d 0 11
prevented n 0 16
preventing g 0 10
prevents z 0 10
price b 0 1
priced n 0 4
pricing g 0 5
pride b 0 1
prides z 0 2
print b 0 4
printed d 0 9
printed n 0 27
printing g 0 13
probe b 0 5
probed d 0 3
probing g 0 5
proceed b 0 18
proceeded d 0 22
proceeded n 0 3
proceeding g 0 7
proceeds z 0 6
process b 0 1
processed n 0 12
processing g 0 21
proclaim b 0 13
proclaimed d 0 5
proclaimed n 0 4
proclaiming g 0 4
proclaims z 0 4
procure b 0 4
procured n 0 4
produce b 0 73
produced d 0 28
produced n 0 62
produces z 0 19
producing g 0 34
profess b 0 5
professed d 0 4
professing g 0 2
proffer b 0 0
proffered d 0 2
profit b 0 4
profited n 0 2
program b 0 5
programed n 0 5
programing g 0 11
progress b 0 4
progressed d 0 10
progressed n 0 3
progresses z 0 6
progressing g 0 2
prohibit b 0 2
prohibited d 0 2
prohibited n 0 6
prohibiting g 0 4
project b 0 12
projected d 0 2
projected n 0 12
projecting g 0 6
projects z 0 4
prolong b 0 1
prolonged n 0 16
prolonging g 0 2
promise b 0 9
promised d 0 26
promised n 0 19

promises	z	0	10
promising	g	0	4
promote	b	0	32
promoted	n	0	11
promotes	z	0	4
promoting	g	0	13
prompt	b	0	2
prompted	d	0	3
prompted	n	0	4
prompts	z	0	2
pronounce	b	0	2
pronounced	d	0	2
pronounced	n	0	16
prop	b	0	2
propped	n	0	3
propel	b	0	4
propose	b	0	13
proposed	d	0	19
proposed	n	0	65
proposes	z	0	7
proposing	g	0	6
prosecute	b	0	2
prosecuted	n	0	5
prosecuting	g	0	3
prosper	b	0	3
protect	b	0	34
protected	d	0	5
protected	n	0	26
protecting	g	0	3
protects	z	0	4
protest	b	0	6
protested	d	0	11
protested	n	0	2
protesting	g	0	7
protests	z	0	3
protrude	b	0	1
protruded	d	0	4
protruding	g	0	3
prove	b	0	53
proved	d	0	48
proved	n	0	23
proves	z	0	16
proving	g	0	5
proven	n	1	11
provide	b	0	216
provided	d	0	29
provided	n	0	103
provides	z	0	81
providing	g	0	56
provoke	b	0	3
provoked	d	0	5
provoked	n	0	2
provokes	z	0	4
pry	b	0	6
publicize	b	0	0
publicized	n	0	6
publicizing	g	0	2
publish	b	0	3
published	d	0	11
published	n	0	78
publishes	z	0	4
publishing	g	0	12
puff	b	0	1
puffed	d	0	2
puffed	n	0	2
puffing	g	0	2
pull	b	0	39
pulled	d	0	54
pulled	n	0	19
pulling	g	0	25
pulls	z	0	8
pump	b	0	1
pumped	d	0	2
pumping	g	0	8
punish	b	0	3
punished	n	0	7
purchase	b	0	11
purchased	d	0	4
purchased	n	0	16
purchases	z	0	2
purchasing	g	0	14
purge	b	0	1
purged	n	0	2
purify	b	0	2
purified	n	0	10
purport	b	0	2
purported	d	0	3
purporting	g	0	2
purports	z	0	2
purse	b	0	0
pursed	d	0	3
pursue	b	0	20
pursued	d	0	3
pursued	n	0	11
pursues	z	0	2
pursuing	g	0	9
push	b	0	30
pushed	d	0	31
pushed	n	0	22
pushes	z	0	2
pushing	g	0	16
put	b	0	197
puts	z	0	20
putting	g	0	54
put	d	1	131
put	n	1	110
puzzle	b	0	4
puzzled	d	0	2
puzzled	n	0	17
qualify	b	0	15
qualified	d	0	2
qualified	n	0	22
qualifies	z	0	2
quarrel	b	0	6
quarreled	d	0	2
quarreled	n	0	2
quarreling	g	0	5
quell	b	0	2
question	b	0	17
questioned	d	0	8
questioned	n	0	21
questioning	g	0	19
questions	z	0	2
quiet	b	0	2
quieted	d	0	2
quit	b	0	12
quit	d	1	2
quitting	g	0	4
quote	b	0	15
quoted	d	0	8
quoted	n	0	18
quotes	z	0	4
quoting	g	0	3
race	b	0	4
raced	d	0	11
racing	g	0	13
rack	b	0	2
radiate	b	0	1
radiated	d	0	2
radiated	n	0	2
rage	b	0	0
raged	d	0	7
raging	g	0	2
rain	b	0	2
rained	d	0	2
rained	n	0	2
raining	g	0	7
raise	b	0	47
raised	d	0	42
raised	n	0	59
raises	z	0	13
raising	g	0	26
rake	b	0	2
raked	d	0	3
rally	b	0	5
ramble	b	0	3
ram	b	0	1
rammed	d	0	3
range	b	0	12
ranged	d	0	16
ranged	n	0	2
ranges	z	0	3
ranging	g	0	30
rank	b	0	0
ranked	d	0	2

ranked	n 0 2	rebelled	n 0 2
ranking	g 0 4	rebelling	g 0 3
ransack	b 0 2	rebuild	b 0 5
ransacked	n 0 2	rebuilding	g 0 5
rape	b 0 3	rebuilt	n 1 4
raped	n 0 2	rebuke	b 0 2
rap	b 0 0	rebut	b 0 6
rapped	d 0 2	recall	b 0 35
rapped	n 0 2	recalled	d 0 19
rapping	g 0 2	recalled	n 0 7
rate	b 0 4	recalling	g 0 5
rated	n 0 9	recalls	z 0 12
rates	z 0 4	recapture	b 0 3
rating	g 0 4	recede	b 0 3
ration	b 0 0	receding	g 0 5
rationed	n 0 3	receive	b 0 76
rationalize	b 0 5	received	d 0 65
rattle	b 0 0	received	n 0 98
rattled	d 0 2	receives	z 0 20
rattling	g 0 6	receiving	g 0 34
reach	b 0 91	recite	b 0 2
reached	d 0 106	reciting	g 0 2
reached	n 0 63	reckon	b 0 7
reaches	z 0 21	reckoned	n 0 3
reaching	g 0 43	reclaim	b 0 2
react	b 0 15	reclaimed	n 0 2
reacted	d 0 12	recognize	b 0 62
reacting	g 0 4	recognized	d 0 30
read	b 0 89	recognized	n 0 50
reading	g 0 86	recognizes	z 0 10
reads	z 0 16	recognizing	g 0 10
read	d 1 36	recommend	b 0 25
read	n 1 47	recommended	d 0 17
readjust	b 0 2	recommended	n 0 29
ready	b 0 2	recommending	g 0 8
realize	b 0 69	recommends	z 0 2
realized	d 0 49	reconcile	b 0 4
realized	n 0 20	reconciled	n 0 3
realizes	z 0 3	reconsider	b 0 4
realizing	g 0 12	reconsidered	n 0 4
reap	b 0 3	reconstruct	b 0 6
rear	b 0 3	reconstructed	n 0 2
reared	d 0 7	record	b 0 11
reared	n 0 3	recorded	d 0 8
rearrange	b 0 3	recorded	n 0 35
reason	b 0 3	recording	g 0 15
reasoned	d 0 4	records	z 0 4
reasoning	g 0 4	recount	b 0 2
reassemble	b 0 3	recounted	d 0 2
reassure	b 0 1	recounting	g 0 3
reassured	d 0 3	recounts	z 0 3
reassured	n 0 2	recover	b 0 11
reassuring	g 0 6	recovered	d 0 2
rebel	b 0 3	recovered	n 0 7
rebelled	d 0 2	recovering	g 0 4

recruit	b 0 9		
recruited	n 0 3		
recruiting	g 0 4		
recur	b 0 2		
recurred	d 0 2		
recurring	g 0 6		
redecorate	b 0 0		
redecorated	n 0 2		
redecorating	g 0 3		
redeem	b 0 2		
reduce	b 0 62		
reduced	d 0 10		
reduced	n 0 69		
reduces	z 0 7		
reducing	g 0 31		
reel	b 0 0		
reeled	d 0 2		
refer	b 0 27		
referred	d 0 14		
referred	n 0 31		
referring	g 0 18		
refers	z 0 18		
refill	b 0 3		
refine	b 0 3		
refined	n 0 6		
refining	g 0 2		
reflect	b 0 25		
reflected	d 0 13		
reflected	n 0 29		
reflecting	g 0 17		
reflects	z 0 23		
reform	b 0 3		
reformed	n 0 3		
refrain	b 0 6		
refuse	b 0 15		
refused	d 0 44		
refused	n 0 16		
refuses	z 0 6		
refusing	g 0 10		
refute	b 0 1		
refuted	d 0 2		
regain	b 0 1		
regained	d 0 3		
regaining	g 0 3		
regard	b 0 36		
regarded	d 0 12		
regarded	n 0 44		
regarding	g 0 3		
regards	z 0 5		
register	b 0 8		
registered	d 0 7		
registered	n 0 16		
registering	g 0 4		
regret	b 0 6		
regrets	z 0 2		

regretted	d	0	7
regretted	n	0	4
regulate	b	0	2
regulated	d	0	2
regulated	n	0	5
regulating	g	0	4
reinforce	b	0	10
reinforced	n	0	6
reinforces	z	0	3
reinforcing	g	0	2
reject	b	0	10
rejected	d	0	12
rejected	n	0	21
rejecting	g	0	4
rejects	z	0	11
rejoin	b	0	2
relate	b	0	7
related	d	0	9
related	n	0	93
relates	z	0	8
relating	g	0	20
relax	b	0	19
relaxed	d	0	6
relaxed	n	0	8
relaxes	z	0	3
relaxing	g	0	5
release	b	0	9
released	d	0	7
released	n	0	19
releases	z	0	2
releasing	g	0	2
relieve	b	0	13
relieved	d	0	3
relieved	n	0	21
relieves	z	0	2
relinquish	b	0	6
relinquished	d	0	3
relinquishing	g	0	3
relish	b	0	3
relive	b	0	2
rely	b	0	13
relying	g	0	5
relied	d	0	2
relied	n	0	6
relies	z	0	4
remain	b	0	93
remained	d	0	84
remained	n	0	22
remaining	g	0	41
remains	z	0	74
remake	b	0	2
remark	b	0	2
remarked	d	0	25
remarked	n	0	7
remarking	g	0	2
remarks	z	0	7
remedy	b	0	3
remember	b	0	138
remembered	d	0	52
remembered	n	0	31
remembering	g	0	16
remembers	z	0	13
remind	b	0	15
reminded	d	0	19
reminded	n	0	10
reminding	g	0	5
reminds	z	0	8
remonstrate	b	0	1
remonstrated	d	0	2
remove	b	0	58
removed	d	0	11
removed	n	0	64
removes	z	0	5
removing	g	0	8
rename	b	0	0
renamed	n	0	3
render	b	0	11
rendered	d	0	5
rendered	n	0	23
rendering	g	0	6
renders	z	0	2
renew	b	0	4
renewed	d	0	3
renewed	n	0	14
rent	b	0	12
rented	n	0	7
renting	g	0	5
repay	b	0	7
repaid	n	0	3
repair	b	0	5
repaired	d	0	2
repaired	n	0	9
repeat	b	0	23
repeated	d	0	18
repeated	n	0	41
repeating	g	0	9
repeats	z	0	4
repel	b	0	8
repelled	n	0	5
repent	b	0	3
replace	b	0	30
replaced	d	0	12
replaced	n	0	30
replaces	z	0	6
replacing	g	0	9
replenish	b	0	4
replenished	d	0	2
reply	b	0	14
replied	d	0	55
replied	n	0	2
replies	z	0	3
report	b	0	39
reported	d	0	51
reported	n	0	68
reporting	g	0	12
reports	z	0	14
represent	b	0	38
represented	d	0	15
represented	n	0	41
representing	g	0	30
represents	z	0	39
reproduce	b	0	7
reproduced	n	0	7
reproduces	z	0	3
reproducing	g	0	2
repute	b	0	0
reputed	n	0	5
request	b	0	7
requested	d	0	5
requested	n	0	7
requesting	g	0	8
requests	z	0	2
require	b	0	86
required	d	0	31
required	n	0	150
requires	z	0	57
requiring	g	0	16
rescind	b	0	2
rescue	b	0	6
rescued	n	0	5
rescuing	g	0	2
resemble	b	0	8
resembled	d	0	8
resembles	z	0	9
resembling	g	0	2
resent	b	0	8
resented	d	0	5
resented	n	0	3
reserve	b	0	7
reserved	d	0	3
reserved	n	0	24
reserving	g	0	4
reside	b	0	2
resided	d	0	3
resides	z	0	4
residing	g	0	6
resign	b	0	2
resigned	d	0	7
resigned	n	0	2
resist	b	0	22
resisted	d	0	4
resisted	n	0	5
resisting	g	0	4
resolve	b	0	11
resolved	d	0	3

resolved	n 0 18	retorted	d 0 3
resolves	z 0 3	retreat	b 0 3
resolving	g 0 3	retreated	d 0 6
resort	b 0 6	retreated	n 0 2
resorted	n 0 4	retreating	g 0 5
resorting	g 0 3	retrieve	b 0 2
respect	b 0 5	retrieved	n 0 9
respected	d 0 5	return	b 0 74
respected	n 0 6	returned	d 0 81
respecting	g 0 3	returned	n 0 34
respects	z 0 3	returning	g 0 35
respond	b 0 21	returns	z 0 8
responded	d 0 13	reveal	b 0 30
responded	n 0 7	revealed	d 0 21
responding	g 0 6	revealed	n 0 18
responds	z 0 7	revealing	g 0 7
rest	b 0 25	reveals	z 0 21
rested	d 0 12	reverse	b 0 11
rested	n 0 5	reversed	n 0 8
resting	g 0 19	reverses	z 0 2
rests	z 0 16	reversing	g 0 6
restore	b 0 9	revert	b 0 3
restored	d 0 3	reverted	d 0 2
restored	n 0 11	review	b 0 10
restoring	g 0 5	reviewed	d 0 3
restrain	b 0 10	reviewed	n 0 10
restrained	n 0 13	reviewing	g 0 10
restraining	g 0 7	reviews	z 0 3
restrict	b 0 11	revise	b 0 5
restricted	n 0 15	revised	d 0 5
restricting	g 0 3	revised	n 0 11
restricts	z 0 2	revive	b 0 8
result	b 0 47	revived	d 0 3
resulted	d 0 32	revived	n 0 3
resulted	n 0 5	reviving	g 0 2
resulting	g 0 43	revolve	b 0 1
results	z 0 17	revolved	d 0 3
resume	b 0 10	revolving	g 0 6
resumed	d 0 12	reward	b 0 2
resumed	n 0 11	rewarded	n 0 3
resuming	g 0 4	rid	b 0 3
retain	b 0 11	ridding	g 0 2
retained	d 0 4	ride	b 0 2
retained	n 0 18	ride	b 0 32
retaining	g 0 7	rides	z 0 4
retains	z 0 9	riding	g 0 42
retard	b 0 3	rode	d 1 40
retarded	n 0 7	ridden	n 1 6
rethink	b 0 2	ridicule	b 0 3
retire	b 0 9	ridiculed	n 0 2
retired	d 0 10	ridiculing	g 0 2
retired	n 0 25	ring	b 0 7
retires	z 0 2	ringing	g 0 8
retiring	g 0 8	rang	d 1 21
retort	b 0 2	rung	n 1 1
ring	b 0 7
ringed	n 0 2
rip	b 0 5
ripped	d 0 5
ripping	g 0 3
rise	b 0 49
rises	z 0 18
risen	n 1 10
rose	d 1 60
rising	g 0 62
risk	b 0 12
risked	d 0 2
risked	n 0 2
rival	b 0 2
rivaled	d 0 2
roam	b 0 6
roaming	g 0 3
roar	b 0 1
roared	d 0 18
roaring	g 0 7
roast	b 0 2
roasted	n 0 4
rob	b 0 2
robbed	d 0 2
robbed	n 0 8
robbing	g 0 2
rock	b 0 3
rocked	d 0 7
rocking	g 0 7
roll	b 0 19
rolled	d 0 34
rolled	n 0 14
rolling	g 0 19
rolls	z 0 2
romanticize	b 0 2
root	b 0 3
rooted	n 0 7
rooting	g 0 2
rot	b 0 2
rots	z 0 3
rotting	g 0 3
rotate	b 0 2
rotated	d 0 2
rotated	n 0 4
rotates	z 0 2
rotating	g 0 11
round	b 0 6
rounded	d 0 5
rounded	n 0 10
rounding	g 0 4
rouse	b 0 2
roused	d 0 2
row	b 0 3
rowed	n 0 2
row	b 0 3

rub	b	0	4	save	b	0	55
rubbed	d	0	13	saved	d	0	11
rubbed	n	0	5	saved	n	0	32
rubbing	g	0	11	saves	z	0	5
ruin	b	0	5	saving	g	0	18
ruined	n	0	16	savor	b	0	0
rule	b	0	7	savored	d	0	2
ruled	d	0	13	savoring	g	0	3
ruled	n	0	18	saw	b	0	9
rules	z	0	3	say	b	0	484
ruling	g	0	12	saying	g	0	112
rumble	b	0	0	says	z	1	197
rumbled	d	0	2	said	d	1	1748
rumbling	g	0	2	said	n	1	214
run	b	0	126	scan	b	0	3
running	g	0	120	scanned	d	0	9
runs	z	0	16	scanning	g	0	3
run	n	1	31	scandalize	b	0	0
ran	d	1	134	scandalized	d	0	2
rush	b	0	3	scare	b	0	2
rushed	d	0	20	scared	d	0	3
rushed	n	0	7	scared	n	0	19
rushes	z	0	2	scatter	b	0	0
rushing	g	0	10	scattered	d	0	7
rustle	b	0	2	scattered	n	0	20
rustling	g	0	9	schedule	b	0	2
sacrifice	b	0	10	scheduled	d	0	2
sacrificed	n	0	2	scheduled	n	0	36
sacrificing	g	0	2	school	b	0	4
safeguard	b	0	2	schooling	g	0	2
sag	b	0	4	scoop	b	0	1
sagged	d	0	3	scooped	d	0	3
sagging	g	0	4	scoot	b	0	0
sail	b	0	6	scooted	d	0	4
sailed	d	0	7	score	b	0	16
sailed	n	0	3	scored	d	0	9
sailing	g	0	16	scored	n	0	6
salt	b	0	0	scoring	g	0	4
salted	n	0	4	scour	b	0	1
salting	g	0	2	scoured	n	0	3
salute	b	0	1	scouring	g	0	3
saluted	d	0	2	scowl	b	0	0
salvage	b	0	2	scowled	d	0	4
salvaging	g	0	2	scowling	g	0	2
sample	b	0	1	scramble	b	0	0
sampled	n	0	6	scrambled	d	0	8
sampling	g	0	12	scrape	b	0	2
sanction	b	0	4	scraped	d	0	6
sanctioned	n	0	4	scraped	n	0	2
sanctions	z	0	2	scraping	g	0	7
satisfy	b	0	16	scratch	b	0	3
satisfied	d	0	5	scratched	d	0	4
satisfied	n	0	31	scratched	n	0	3
satisfies	z	0	3	scratching	g	0	12
satisfying	g	0	8	scrawl	b	0	0
scrawled	d	0	3
scrawled	n	0	2
scream	b	0	7
screamed	d	0	14
screamed	n	0	3
screaming	g	0	16
screech	b	0	0
screeched	d	0	5
screeching	g	0	7
screen	b	0	5
scrub	b	0	6
scrubbing	g	0	2
scrutinize	b	0	0
scrutinized	d	0	2
scrutinizing	g	0	3
scurry	b	0	0
scurried	d	0	3
seal	b	0	3
sealed	n	0	13
sealing	g	0	3
sear	b	0	2
searing	g	0	2
search	b	0	8
searched	d	0	7
searched	n	0	2
searches	z	0	2
searching	g	0	22
seat	b	0	1
seated	d	0	2
seated	n	0	23
seating	g	0	5
secede	b	0	10
seceded	n	0	2
seceding	g	0	2
secure	b	0	16
secured	n	0	10
securing	g	0	6
see	b	0	771
seeing	g	0	85
sees	z	0	36
seen	n	1	279
saw	d	1	337
seek	b	0	69
seeking	g	0	44
seeks	z	0	10
sought	d	1	35
sought	n	1	20
seem	b	0	229
seemed	d	0	311
seemed	n	0	22
seeming	g	0	10
seems	z	0	259
seep	b	0	2
seeping	g	0	2
seize	b	0	6

seized	d 0 12	shaking	g 0 20
seized	n 0 12	shaken	n 1 11
select	b 0 18	shook	d 1 57
selected	d 0 8	shape	b 0 6
selected	n 0 66	shaped	d 0 3
selecting	g 0 15	shaped	n 0 14
selects	z 0 5	shapes	z 0 4
sell	b 0 39	shaping	g 0 7
selling	g 0 28	share	b 0 40
sells	z 0 13	shared	d 0 19
sold	d 1 20	shared	n 0 21
sold	n 1 27	shares	z 0 4
send	b 0 74	sharing	g 0 21
sending	g 0 30	shatter	b 0 2
sends	z 0 4	shattered	d 0 6
sent	d 1 69	shattered	n 0 7
sent	n 1 76	shattering	g 0 6
sense	b 0 10	shave	b 0 6
sensed	d 0 16	shaved	d 0 4
senses	z 0 3	shaved	n 0 5
sensing	g 0 5	shaving	g 0 6
separate	b 0 14	shaven	n 1 2
separated	d 0 6	shear	b 0 2
separated	n 0 37	shearing	g 0 4
separates	z 0 3	shed	b 0 2
separating	g 0 7	shedding	g 0 2
serve	b 0 107	sheds	z 0 3
served	d 0 53	shed	d 1 3
served	n 0 66	shed	n 1 2
serves	z 0 37	shield	b 0 2
serving	g 0 37	shielded	n 0 5
service	b 0 4	shift	b 0 15
servicing	g 0 2	shifted	d 0 12
set	b 0 92	shifted	n 0 6
sets	z 0 15	shifting	g 0 12
setting	g 0 32	shifts	z 0 2
set	d 1 71	shimmy	b 0 2
set	n 1 161	shine	b 0 2
settle	b 0 23	shines	z 0 4
settled	d 0 31	shining	g 0 21
settled	n 0 38	shone	d 1 5
settles	z 0 2	ship	b 0 6
settling	g 0 11	shipped	n 0 5
sever	b 0 3	shipping	g 0 15
severed	n 0 5	shiver	b 0 2
sew	b 0 6	shivered	d 0 3
sewing	g 0 10	shivering	g 0 11
shade	b 0 2	shock	b 0 2
shaded	n 0 5	shocked	d 0 2
shading	g 0 2	shocked	n 0 17
shadow	b 0 2	shoot	b 0 26
shadowed	n 0 3	shooting	g 0 36
shadowing	g 0 4	shot	d 1 18
shake	b 0 15	shot	n 1 34
shakes	z 0 4	shop	b 0 7

shopping	g 0 19	shouted	d 0 36
shorten	b 0 4	shouted	n 0 4
shortened	n 0 7	shouting	g 0 29
shoulder	b 0 1	shouts	z 0 3
shouldered	d 0 3	shove	b 0 2
shout	b 0 5	shoved	d 0 8
shoving	g 0 6	shrilled	d 0 2
show	b 0 202	shrink	b 0 5
showing	g 0 57	shrinking	g 0 3
shows	z 0 73	shrinks	z 0 2
showed	d 0 138	shrunken	n 1 1
showed	n 0 3	shrug	b 0 0
shown	n 1 166	shrugged	d 0 18
shower	b 0 2	shudder	b 0 5
showered	d 0 4	shuddered	d 0 5
shred	b 0 2	shuddering	g 0 3
shriek	b 0 1	shuffle	b 0 1
shrieked	d 0 4	shuffled	d 0 2
shrill	b 0 0	shuffling	g 0 3
		shun	b 0 1
		shuns	z 0 2
		shut	b 0 15
		shut	d 1 7
		shut	n 1 24
		shy	b 0 2
		shied	d 0 3
		sicken	b 0 0
		sickened	n 0 3
		sickening	g 0 2
		sidle	b 0 1
		sidled	d 0 2
		sift	b 0 0
		sifted	d 0 2
		sigh	b 0 1
		sighed	d 0 22
		sighing	g 0 5

sight	b 0 1	sketches	z 0 2
sighted	d 0 2	sketching	g 0 4
sighted	n 0 5	skid	b 0 0
sighting	g 0 2	skidded	d 0 2
sign	b 0 18	skidding	g 0 2
signed	d 0 15	skim	b 0 0
signed	n 0 22	skimmed	d 0 2
signing	g 0 5	skimmed	n 0 3
signs	z 0 2	skimming	g 0 3
signal	b 0 4	skip	b 0 4
signaled	d 0 2	skipped	d 0 6
signaling	g 0 3	skipped	n 0 2
signify	b 0 2	skipping	g 0 4
silence	b 0 1	skirt	b 0 2
silenced	d 0 2	slacken	b 0 0
silenced	n 0 3	slackened	n 0 2
simmer	b 0 5	slackening	g 0 2
simplify	b 0 9	slam	b 0 0
simplified	n 0 9	slammed	d 0 13
simplifies	z 0 2	slammed	n 0 3
simulate	b 0 4	slamming	g 0 4
simulated	n 0 7	slant	b 0 1
sin	b 0 2	slanted	n 0 2
sinned	n 0 6	slanting	g 0 4
sing	b 0 27	slap	b 0 2
sing	b 0 4	slapped	d 0 6
singing	g 0 33	slapped	n 0 2
sings	z 0 10	slapping	g 0 6
sang	d 1 28	slash	b 0 3
sung	n 1 18	slashed	d 0 6
single	b 0 1	slashed	n 0 4
singled	d 0 7	slashing	g 0 5
singled	n 0 3	sleep	b 0 31
sink	b 0 11	sleeping	g 0 38
sinking	g 0 5	slept	d 1 18
sank	d 1 18	slept	n 1 9
sunk	n 1 6	slice	b 0 3
sip	b 0 0	sliced	d 0 3
sipped	d 0 2	slide	b 0 8
sipping	g 0 8	sliding	g 0 11
sit	b 0 66	slid	d 1 24
sits	z 0 6	slide	b 0 8
sitting	g 0 92	sling	b 0 0
sat	d 1 139	slung	n 1 1
sat	n 1 11	slung	d 1 1
size	b 0 2	slip	b 0 7
sized	d 0 2	slipped	d 0 26
sizzle	b 0 1	slipped	n 0 6
sizzled	d 0 2	slipping	g 0 7
sizzling	g 0 2	slit	b 0 2
skate	b 0 1	slow	b 0 8
skating	g 0 2	slowed	d 0 12
sketch	b 0 1	slowed	n 0 5
sketched	d 0 2	slowing	g 0 5
sketched	n 0 2	slug	b 0 2
slugged	d 0 3		
slugging	g 0 3		
slump	b 0 3		
slumped	d 0 6		
slumped	n 0 2		
smack	b 0 4		
smacked	d 0 2		
smash	b 0 2		
smashed	d 0 11		
smashed	n 0 4		
smell	b 0 9		
smelled	d 0 15		
smelled	n 0 4		
smelling	g 0 5		
smells	z 0 7		
smile	b 0 10		
smiled	d 0 68		
smiled	n 0 3		
smiles	z 0 3		
smiling	g 0 36		
smoke	b 0 8		
smoked	d 0 6		
smoked	n 0 3		
smoking	g 0 8		
smolder	b 0 0		
smoldered	d 0 2		
smoldering	g 0 2		
smooth	b 0 6		
smoothed	d 0 4		
smoothed	n 0 3		
smoothing	g 0 2		
smother	b 0 0		
smothered	d 0 2		
smothered	n 0 4		
snag	b 0 3		
snake	b 0 0		
snaked	d 0 3		
snap	b 0 11		
snapped	d 0 17		
snapped	n 0 2		
snapping	g 0 8		
snarl	b 0 0		
snarled	d 0 8		
snarling	g 0 3		
snatch	b 0 4		
snatched	d 0 9		
snatched	n 0 2		
sneak	b 0 1		
sneaked	d 0 4		
sneaked	n 0 2		
sneaking	g 0 2		
snicker	b 0 0		
snickered	d 0 2		
sniff	b 0 2		
sniffed	d 0 6		

sniffing	g	0	2
snort	b	0	2
snorted	d	0	4
snow	b	0	1
snowed	d	0	2
snowing	g	0	4
snows	z	0	5
snuggle	b	0	0
snuggled	d	0	4
soak	b	0	7
soaked	n	0	5
soaking	g	0	5
soar	b	0	0
soared	d	0	3
soaring	g	0	5
sober	b	0	0
sobered	n	0	3
sobering	g	0	3
soften	b	0	4
softened	d	0	3
softened	n	0	4
softening	g	0	4
soil	b	0	2
soiled	n	0	7
solder	b	0	3
solve	b	0	20
solved	n	0	18
solves	z	0	2
solving	g	0	8
soothe	b	0	2
soothed	d	0	2
soothing	g	0	4
sort	b	0	5
sorted	n	0	3
sound	b	0	25
sounded	d	0	29
sounded	n	0	6
sounding	g	0	2
sounds	z	0	20
sow	b	0	3
sown	n	1	3
space	b	0	1
spaced	n	0	8
spacing	g	0	2
span	b	0	4
spanned	n	0	2
spans	z	0	3
spare	b	0	8
spared	d	0	3
spared	n	0	6
spark	b	0	2
sparked	n	0	2
sparkling	g	0	4
sparks	z	0	2
speak	b	0	110
speaking	g	0	59
speaks	z	0	18
spoke	d	1	86
spoken	n	1	37
spear	b	0	2
specialize	b	0	3
specialized	d	0	2
specialized	n	0	16
specializing	g	0	4
specify	b	0	11
specified	n	0	28
specifies	z	0	4
specifying	g	0	3
speculate	b	0	7
speculated	d	0	2
speculating	g	0	4
speed	b	0	5
speeding	g	0	4
sped	d	1	9
spell	b	0	5
spelled	n	0	5
spells	z	0	2
spend	b	0	53
spending	g	0	29
spends	z	0	8
spent	d	1	40
spent	n	1	64
spice	b	0	2
spiced	n	0	3
spill	b	0	1
spilled	d	0	2
spilling	g	0	3
spills	z	0	2
spin	b	0	4
spinning	g	0	11
spun	d	1	14
spun	n	1	2
spit	b	0	6
spitting	g	0	5
spat	d	1	7
splash	b	0	0
splashed	d	0	3
splashing	g	0	3
split	b	0	2
splitting	g	0	3
split	d	1	5
split	n	1	15
spoil	b	0	3
spoiled	n	0	5
sponge	b	0	2
sponged	n	0	2
sponsor	b	0	7
sponsored	d	0	7
sponsored	n	0	23
sponsoring	g	0	3
sponsors	z	0	3
spot	b	0	4
spotted	d	0	9
spotted	n	0	7
spotting	g	0	2
spout	b	0	0
spouted	d	0	2
sprawl	b	0	2
sprawled	d	0	2
sprawled	n	0	9
sprawling	g	0	5
spray	b	0	2
sprayed	d	0	3
sprayed	n	0	3
spraying	g	0	6
spread	b	0	27
spreading	g	0	15
spreads	z	0	9
spread	d	1	18
spread	n	1	21
spring	b	0	6
springing	g	0	2
sprang	d	1	13
sprung	n	1	8
sprinkle	b	0	5
sprinkled	d	0	3
sprinkling	g	0	2
sprint	b	0	0
sprinted	d	0	2
sprout	b	0	1
sprouted	n	0	2
sprouting	g	0	4
spur	b	0	3
spurred	d	0	4
spurred	n	0	2
square	b	0	9
squared	n	0	5
squatted	d	0	4
squeak	b	0	0
squeaked	d	0	2
squeeze	b	0	8
squeezed	d	0	8
squeezed	n	0	10
squeezing	g	0	4
squint	b	0	0
squinted	d	0	3
squinting	g	0	3
stabilize	b	0	2
stabilizing	g	0	4
stack	b	0	0
stacked	n	0	8
stacking	g	0	2
staff	b	0	2
staffed	n	0	3
stage	b	0	2

staged	d	0	5
staged	n	0	11
staging	g	0	3
stagger	b	0	2
staggered	d	0	9
staggered	n	0	3
staggering	g	0	3
stain	b	0	2
stained	d	0	4
stained	n	0	24
staining	g	0	14
stake	b	0	2
stalk	b	0	0
stalked	d	0	6
stalking	g	0	2
stall	b	0	0
stalled	d	0	3
stalling	g	0	2
stammer	b	0	0
stammered	d	0	4
stamp	b	0	5
stamped	d	0	2
stamped	n	0	5
stamping	g	0	4
stampede	b	0	2
stand	b	0	109
standing	g	0	96
stands	z	0	49
stood	d	1	198
stood	n	1	14
stare	b	0	9
stared	d	0	58
stared	n	0	2
staring	g	0	26
star	b	0	0
starred	n	0	2
starring	g	0	2
start	b	0	103
started	d	0	139
started	n	0	55
starts	z	0	21
starting	g	0	67
startle	b	0	1
startled	d	0	3
startled	n	0	18
starve	b	0	1
starved	n	0	3
starving	g	0	6
state	b	0	20
stated	d	0	38
stated	n	0	47
states	z	0	17
stating	g	0	16
stave	b	0	2
stay	b	0	99
stayed	d	0	60
stayed	n	0	15
staying	g	0	17
stays	z	0	3
steady	b	0	1
steadied	d	0	2
steal	b	0	5
stealing	g	0	4
stole	d	1	10
stolen	n	1	18
steam	b	0	0
steamed	n	0	5
steaming	g	0	5
steer	b	0	3
steered	d	0	4
steering	g	0	9
stem	b	0	5
stemmed	d	0	3
stems	z	0	14
step	b	0	21
stepped	d	0	33
stepped	n	0	7
stepping	g	0	9
stick	b	0	16
sticking	g	0	8
sticks	z	0	3
stuck	d	1	13
stuck	n	1	10
stiffen	b	0	0
stiffened	d	0	7
stiffens	z	0	2
stifle	b	0	2
stifling	g	0	2
still	b	0	2
stimulate	b	0	6
stimulated	d	0	3
stimulated	n	0	4
stimulates	z	0	3
stimulating	g	0	4
sting	b	0	2
stinging	g	0	2
stung	d	1	1
stung	n	1	1
stink	b	0	1
stipulate	b	0	2
stipulates	z	0	2
stir	b	0	7
stirred	d	0	7
stirred	n	0	8
stirring	g	0	13
stirs	z	0	3
stock	b	0	2
stoop	b	0	3
stooped	d	0	3
stooping	g	0	4
stop	b	0	94
stopped	d	0	103
stopped	n	0	26
stopping	g	0	14
stops	z	0	3
store	b	0	7
stored	n	0	35
storing	g	0	4
storm	b	0	1
stormed	d	0	3
straggle	b	0	2
straggling	g	0	2
straighten	b	0	7
straightened	d	0	15
straightened	n	0	4
straightening	g	0	6
strain	b	0	1
strained	d	0	8
strained	n	0	3
straining	g	0	7
stray	b	0	4
streak	b	0	0
streaked	d	0	3
stream	b	0	2
streamed	d	0	2
streaming	g	0	7
strengthen	b	0	16
strengthened	n	0	5
strengthening	g	0	8
strengthens	z	0	5
stress	b	0	11
stressed	d	0	11
stressed	n	0	12
stresses	z	0	4
stressing	g	0	5
stretch	b	0	7
stretched	d	0	21
stretched	n	0	13
stretches	z	0	4
stretching	g	0	16
stride	b	0	4
strode	d	1	10
strike	b	0	25
strikes	z	0	8
striking	g	0	10
stricken	n	1	6
struck	d	1	40
struck	n	1	19
string	b	0	1
strung	d	1	1
strung	n	1	3
strip	b	0	5
stripped	d	0	7
stripped	n	0	10
strive	b	0	7

strives	z 0 3	sucked	d 0 5
striving	g 0 3	sucking	g 0 8
strove	d 1 4	sue	b 0 13
striven	n 1 1	sued	d 0 2
stroke	b 0 1	sued	n 0 2
stroked	d 0 2	suffer	b 0 33
stroll	b 0 0	suffered	d 0 22
strolled	d 0 4	suffered	n 0 20
strolling	g 0 4	suffering	g 0 30
struggle	b 0 7	suffers	z 0 5
struggled	d 0 7	suffice	b 0 5
struggling	g 0 20	suffuse	b 0 1
strut	b 0 1	suffused	d 0 4
strutted	d 0 2	suggest	b 0 54
study	b 0 41	suggested	d 0 48
studying	g 0 40	suggested	n 0 56
studied	d 0 34	suggesting	g 0 13
studied	n 0 45	suggests	z 0 29
stuff	b 0 3	suit	b 0 8
stuffed	n 0 4	suited	d 0 3
stuffing	g 0 2	suited	n 0 19
stumble	b 0 1	sulk	b 0 0
stumbled	d 0 18	sulked	d 0 2
stumbled	n 0 3	sum	b 0 1
stumbling	g 0 8	sums	z 0 2
stun	b 0 0	summed	d 0 4
stunned	n 0 7	summed	n 0 3
subdue	b 0 2	summarize	b 0 3
subdued	n 0 8	summarized	n 0 8
submit	b 0 18	summarizing	g 0 3
submits	z 0 3	summate	b 0 2
submitted	d 0 10	summon	b 0 3
submitted	n 0 11	summoned	d 0 5
submitting	g 0 5	summoned	n 0 5
subscribe	b 0 1	superimpose	b 0 4
subscribed	n 0 2	superimposed	n 0 6
subscribing	g 0 2	supervise	b 0 5
subside	b 0 2	supervised	d 0 3
subsided	d 0 4	supervises	z 0 2
subsidize	b 0 4	supervising	g 0 3
subsidized	n 0 3	supplant	b 0 3
substantiate	b 0 2	supplement	b 0 7
substitute	b 0 5	supplemented	d 0 2
substituted	d 0 3	supplemented	n 0 4
substituted	n 0 12	supplementing	g 0 3
substituting	g 0 4	supply	b 0 43
subtract	b 0 2	supplying	g 0 13
subtracted	n 0 3	supplied	d 0 9
subtracting	g 0 3	supplied	n 0 27
succeed	b 0 15	supplies	z 0 11
succeeded	d 0 18	support	b 0 54
succeeded	n 0 15	supported	d 0 17
succeeding	g 0 5	supported	n 0 37
succeeds	z 0 8	supporting	g 0 27
suck	b 0 4	supports	z 0 9
suppose	b 0 98		
supposed	d 0 8		
supposed	n 0 57		
suppress	b 0 6		
suppressed	n 0 3		
surge	b 0 1		
surged	d 0 7		
surging	g 0 2		
surprise	b 0 5		
surprised	d 0 9		
surprised	n 0 49		
surprising	g 0 13		
surrender	b 0 7		
surrendered	d 0 6		
surrendering	g 0 4		
surround	b 0 5		
surrounded	d 0 4		
surrounded	n 0 17		
surrounding	g 0 27		
survey	b 0 3		
surveyed	d 0 2		
surveyed	n 0 7		
surveying	g 0 8		
survive	b 0 33		
survived	d 0 5		
survived	n 0 9		
surviving	g 0 14		
suspect	b 0 20		
suspected	d 0 12		
suspected	n 0 9		
suspecting	g 0 2		
suspects	z 0 4		
suspend	b 0 3		
suspended	n 0 29		
sustain	b 0 14		
sustained	n 0 14		
sustaining	g 0 2		
swagger	b 0 0		
swaggered	d 0 3		
swallow	b 0 5		
swallowed	d 0 6		
swallowed	n 0 6		
swallowing	g 0 3		
swap	b 0 2		
swarm	b 0 1		
swarmed	d 0 3		
swarming	g 0 3		
sway	b 0 1		
swayed	d 0 7		
swayed	n 0 2		
swaying	g 0 3		
swear	b 0 10		
swearing	g 0 2		
swears	z 0 2		
swore	d 1 14		

sworn	n 1 5	taper	b 0 1
sweep	b 0 7	tapered	n 0 7
sweeping	g 0 13	tapering	g 0 2
swell	b 0 3	taste	b 0 5
swelled	d 0 3	tasted	d 0 7
swelling	g 0 10	tasted	n 0 3
swollen	n 1 4	tastes	z 0 4
sweep	b 0 7	tasting	g 0 3
swept	d 1 19	tax	b 0 5
swept	n 1 15	taxed	n 0 12
swerve	b 0 1	taxing	g 0 9
swerved	d 0 2	teach	b 0 41
swerving	g 0 2	teaches	z 0 11
swim	b 0 10	teaching	g 0 51
swimming	g 0 37	taught	d 1 19
swam	d 1 6	taught	n 1 31
swum	n 1 1	team	b 0 0
swing	b 0 11	teamed	d 0 2
swinging	g 0 16	tear	b 0 9
swung	d 1 43	tearing	g 0 9
swung	n 1 5	tore	d 1 15
swirl	b 0 0	torn	n 1 25
swirled	d 0 5	tease	b 0 4
swirling	g 0 4	teased	d 0 2
swish	b 0 0	teasing	g 0 2
swished	d 0 3	telegraph	b 0 1
switch	b 0 5	telegraphed	d 0 2
switched	d 0 10	telephone	b 0 3
switched	n 0 6	telephoned	d 0 13
switching	g 0 6	telephoned	n 0 3
swoop	b 0 0	telephoning	g 0 2
swooped	d 0 4	tell	b 0 268
swooping	g 0 3	telling	g 0 43
symbolize	b 0 9	tells	z 0 34
symbolized	d 0 6	told	d 1 286
symbolized	n 0 5	told	n 1 127
symbolizes	z 0 3	tempt	b 0 2
symbolizing	g 0 2	tempted	n 0 13
sympathize	b 0 7	tempting	g 0 2
tackle	b 0 4	tend	b 0 43
take	b 0 606	tended	d 0 15
takes	z 0 86	tended	n 0 9
taking	g 0 170	tending	g 0 3
taken	n 1 281	tends	z 0 34
took	d 1 426	term	b 0 3
talk	b 0 114	termed	d 0 3
talked	d 0 41	termed	n 0 12
talked	n 0 17	terms	z 0 3
talking	g 0 99	terminate	b 0 12
talks	z 0 3	terminated	n 0 4
tally	b 0 2	terrify	b 0 0
tap	b 0 10	terrified	n 0 7
tapped	d 0 2	terrifies	z 0 2
tapped	n 0 5	terrifying	g 0 5
tapping	g 0 2	test	b 0 18
tested	d 0 3	thank	b 0 36
tested	n 0 34	thanked	d 0 5
testing	g 0 11	thanking	g 0 3
testify	b 0 8	thaw	b 0 3
testified	d 0 9	thawed	n 0 3
testified	n 0 2	thawing	g 0 2
testifies	z 0 4	theorize	b 0 2
thicken	b 0 1		
thickened	d 0 2		
thickened	n 0 3		
thin	b 0 2		
think	b 0 433		
thinking	g 0 105		
thinks	z 0 23		
thought	d 1 340		
thought	n 1 74		
thrash	b 0 1		
thrashed	d 0 2		
threaten	b 0 11		
threatened	d 0 15		
threatened	n 0 14		
threatening	g 0 22		
threatens	z 0 5		
thrill	b 0 1		
thrilled	d 0 2		
thrive	b 0 1		
thrived	d 0 4		
throb	b 0 0		
throbbed	d 0 3		
throbbing	g 0 2		
throw	b 0 35		
throwing	g 0 17		
throws	z 0 6		
threw	d 1 46		
thrown	n 1 40		
thrust	b 0 3		
thrusting	g 0 7		
thrust	d 1 9		
thrust	n 1 3		
thunder	b 0 2		
thundered	d 0 2		
thundering	g 0 2		
thwart	b 0 2		
thwarted	n 0 4		
tick	b 0 2		
ticked	d 0 2		
tie	b 0 9		
tied	d 0 13		

tied	n 0 21	tracking	g 0 3
ties	z 0 2	trade	b 0 13
tying	g 0 5	traded	d 0 3
tighten	b 0 3	traded	n 0 5
tightened	d 0 6	trading	g 0 25
tightening	g 0 3	trail	b 0 2
tilt	b 0 2	trailed	d 0 6
tilted	d 0 6	trailed	n 0 2
tilted	n 0 6	trailing	g 0 7
tilts	z 0 2	train	b 0 10
time	b 0 1	trained	d 0 2
timed	d 0 2	trained	n 0 52
timed	n 0 7	training	g 0 2
timing	g 0 5	training	g 0 63
tip	b 0 2	tramp	b 0 0
tipped	d 0 4	tramped	d 0 2
tire	b 0 2	trample	b 0 3
tired	d 0 4	transact	b 0 3
tired	n 0 35	transcend	b 0 1
tiring	g 0 4	transcending	g 0 3
toast	b 0 5	transcends	z 0 6
toasting	g 0 2	transfer	b 0 6
toe	b 0 2	transferred	d 0 2
tolerate	b 0 4	transferred	n 0 27
tolerated	n 0 6	transform	b 0 7
top	b 0 3	transformed	d 0 4
topped	d 0 3	transformed	n 0 21
topped	n 0 4	transforming	g 0 2
topping	g 0 2	transforms	z 0 3
torment	b 0 0	translate	b 0 15
tormented	n 0 3	translated	n 0 15
tormenting	g 0 2	translating	g 0 2
toss	b 0 6	transmit	b 0 3
tossed	d 0 22	transmitted	n 0 8
tossed	n 0 9	transpire	b 0 0
tossing	g 0 4	transpired	n 0 2
total	b 0 6	transpiring	g 0 4
totaled	d 0 6	transplant	b 0 2
totaling	g 0 6	transport	b 0 3
totalled	d 0 3	transported	n 0 4
totals	z 0 3	transporting	g 0 3
touch	b 0 32	trap	b 0 0
touched	d 0 24	trapped	n 0 7
touched	n 0 18	trapping	g 0 2
touches	z 0 5	travel	b 0 28
touching	g 0 12	traveled	d 0 13
tour	b 0 1	traveled	n 0 9
toured	d 0 2	traveling	g 0 19
touring	g 0 7	traverse	b 0 5
trace	b 0 7	traversed	d 0 2
traced	d 0 2	traversed	n 0 4
traced	n 0 10	tread	b 0 2
tracing	g 0 16	treat	b 0 24
track	b 0 2	treated	d 0 11
tracked	d 0 2	treated	n 0 64
treating	g 0 11		
treats	z 0 12		
tremble	b 0 10		
trembled	d 0 5		
trembling	g 0 26		
trim	b 0 11		
trimmed	n 0 4		
trip	b 0 1		
tripped	d 0 2		
tripping	g 0 2		
trot	b 0 6		
trotted	d 0 5		
trotted	n 0 2		
trouble	b 0 4		
troubled	d 0 8		
troubled	n 0 23		
troubling	g 0 2		
trudge	b 0 0		
trudged	d 0 4		
trust	b 0 25		
trusted	d 0 2		
trusted	n 0 9		
trusting	g 0 2		
trusts	z 0 2		
try	b 0 137		
trying	g 0 155		
tried	d 0 120		
tried	n 0 50		
tries	z 0 8		
tuck	b 0 1		
tucked	d 0 4		
tucked	n 0 2		
tucking	g 0 2		
tug	b 0 1		
tugged	d 0 2		
tumble	b 0 1		
tumbled	d 0 7		
tumbled	n 0 6		
tumbling	g 0 3		
tune	b 0 2		
tuned	n 0 3		
turn	b 0 145		
turned	d 0 253		
turned	n 0 67		
turning	g 0 70		
turns	z 0 30		
twine	b 0 0		
twined	d 0 3		
twist	b 0 5		
twisted	d 0 12		
twisted	n 0 7		
twisting	g 0 10		
twitch	b 0 0		
twitched	d 0 4		
twitching	g 0 2		

type	b 0 2	unscrew	b 0 1
typed	n 0 2	unscrewed	d 0 2
typing	g 0 7	untie	b 0 2
uncover	b 0 4	upgrade	b 0 3
uncovered	n 0 7	uphold	b 0 7
underestimate	b 0 4	upholding	g 0 4
underestimated	n 0 2	upheld	d 1 5
undergo	b 0 8	upheld	n 1 1
undergoes	z 0 2	upset	b 0 2
undergoing	g 0 12	upsets	z 0 2
underwent	d 1 2	upset	n 1 10
underlie	b 0 2	upset	d 1 10
underlying	g 0 20	urge	b 0 13
underline	b 0 2	urged	d 0 21
underlined	n 0 2	urged	n 0 14
underlining	g 0 2	urges	z 0 6
undermine	b 0 8	urging	g 0 10
undermined	n 0 2	use	b 0 230
underscore	b 0 1	used	d 0 137
underscored	n 0 2	used	n 0 474
understand	b 0 137	uses	z 0 32
understanding	g 0 38	using	g 0 143
understands	z 0 6	usher	b 0 1
understood	d 1 20	ushered	d 0 2
understood	n 1 38	utilize	b 0 10
undertake	b 0 13	utilized	n 0 9
undertakes	z 0 3	utilizes	z 0 4
undertaken	n 1 18	utilizing	g 0 8
underwrite	b 0 3	utter	b 0 3
undo	b 0 3	uttered	d 0 3
undone	n 1 4	uttered	n 0 2
undid	d 1 1	uttering	g 0 2
undress	b 0 0	validate	b 0 2
undressed	d 0 2	value	b 0 1
undressing	g 0 3	valued	n 0 13
unfold	b 0 2	values	z 0 3
unfolded	d 0 2	vanish	b 0 5
unfolded	n 0 2	vanished	d 0 9
unfolding	g 0 5	vanished	n 0 6
unfolds	z 0 4	vanishing	g 0 4
unify	b 0 2	vary	b 0 34
unified	n 0 11	varied	d 0 7
unifies	z 0 3	varied	n 0 35
unifying	g 0 4	varies	z 0 11
unite	b 0 10	varying	g 0 40
united	d 0 3	veer	b 0 2
united	n 0 479	veered	d 0 3
uniting	g 0 2	veering	g 0 2
unload	b 0 7	vent	b 0 2
unloaded	n 0 5	venture	b 0 6
unloading	g 0 5	ventured	d 0 2
unlock	b 0 3	ventured	n 0 3
unlocked	d 0 5	verify	b 0 5
unlocked	n 0 7	verified	n 0 6
unlocks	z 0 2	vex	b 0 1
vexed	n 0 2	wailing	g 0 5
vexing	g 0 2	wait	b 0 83
view	b 0 18	waited	d 0 68
viewed	d 0 2	waited	n 0 2
viewed	n 0 23	waiting	g 0 107
viewing	g 0 10	waits	z 0 2
views	z 0 2	wake	b 0 16
violate	b 0 7	wakes	z 0 2
violated	d 0 2	waking	g 0 11
violated	n 0 2	woke	d 1 14
violates	z 0 2	walk	b 0 66
violating	g 0 4		
visit	b 0 50		
visited	d 0 24		
visited	n 0 17		
visiting	g 0 36		
visits	z 0 2		
visualize	b 0 3		
voice	b 0 0		
voiced	d 0 4		
volunteer	b 0 4		
volunteered	d 0 2		
volunteered	n 0 3		
volunteering	g 0 2		
vote	b 0 26		
voted	d 0 22		
voted	n 0 5		
votes	z 0 3		
voting	g 0 23		
vow	b 0 0		
vowed	d 0 5		
vowing	g 0 2		
wad	b 0 0		
wadded	d 0 2		
wade	b 0 2		
waded	d 0 2		
wag	b 0 0		
wagged	d 0 2		
wagging	g 0 2		
wage	b 0 3		
waged	n 0 6		
wager	b 0 2		
wail	b 0 0		
wailed	d 0 3		

walked	d	0	143
walked	n	0	16
walking	g	0	54
walks	z	0	7
wall	b	0	2
wander	b	0	8
wandered	d	0	7
wandering	g	0	6
wanders	z	0	2
want	b	0	319
wanted	d	0	205
wanted	n	0	21
wanting	g	0	16
wants	z	0	64
warm	b	0	3
warmed	d	0	6
warmed	n	0	4
warming	g	0	9
warn	b	0	11
warned	d	0	14
warned	n	0	8
warning	g	0	26
warns	z	0	3
warp	b	0	0
warped	n	0	3
warping	g	0	4
warrant	b	0	11
warranted	n	0	2
wash	b	0	10
washed	d	0	10
washed	n	0	25
washing	g	0	38
waste	b	0	10
wasted	d	0	5
wasted	n	0	11
wasting	g	0	5
watch	b	0	53
watched	d	0	68
watched	n	0	13
watching	g	0	74
water	b	0	1
watered	d	0	4
watered	n	0	3
watering	g	0	4
wave	b	0	2
waved	d	0	16
waving	g	0	12
waver	b	0	2
wax	b	0	1
waxed	d	0	3
weaken	b	0	7
weakened	n	0	5
wear	b	0	32
wearing	g	0	47
wears	z	0	6
wore	d	1	65
worn	n	1	23
weave	b	0	3
weaving	g	0	5
wove	d	1	3
woven	n	1	9
weep	b	0	14
weeping	g	0	5
wept	d	1	7
wept	n	1	2
weigh	b	0	4
weighed	d	0	11
weighed	n	0	5
weighing	g	0	9
weighs	z	0	4
welcome	b	0	15
welcomed	d	0	6
welcomed	n	0	6
welcoming	g	0	5
well	b	0	4
wet	b	0	3
wetting	g	0	3
wet	d	1	2
wet	n	1	1
whack	b	0	0
whacked	d	0	2
wheel	b	0	0
wheeled	d	0	7
wheeled	n	0	3
while	b	0	2
whine	b	0	2
whining	g	0	6
whinny	b	0	0
whinnied	d	0	2
whip	b	0	5
whipped	d	0	7
whipped	n	0	5
whipping	g	0	7
whirl	b	0	2
whirled	d	0	6
whirling	g	0	9
whisper	b	0	4
whispered	d	0	20
whispered	n	0	3
whispering	g	0	4
whistle	b	0	1
whistled	d	0	6
whistling	g	0	5
whiz	b	0	0
whizzed	d	0	2
widen	b	0	5
widened	d	0	2
widened	n	0	3
wield	b	0	1
wielded	d	0	2
wiggle	b	0	0
wiggled	d	0	3
wiggling	g	0	2
will	b	0	1
willed	d	0	3
willed	n	0	3
win	b	0	54
wins	z	0	5
winning	g	0	31
wince	b	0	0
winced	d	0	4
wind	b	0	7
winding	g	0	9
winds	z	0	3
wound	d	1	7
wound	n	1	3
wind	b	0	7
winded	n	0	2
wing	b	0	1
winged	d	0	2
wink	b	0	3
winked	d	0	7
winking	g	0	7
wipe	b	0	10
wiped	d	0	11
wiped	n	0	8
wiping	g	0	6
wire	b	0	0
wired	d	0	4
wired	n	0	7
wish	b	0	87
wished	d	0	52
wished	n	0	4
wishes	z	0	13
wishing	g	0	5
withdraw	b	0	8
withdrawing	g	0	4
withdrawn	n	1	4
withdrew	d	1	9
wither	b	0	2
withered	n	0	2
withhold	b	0	2
withheld	n	1	7
withheld	d	1	1
withholding	g	0	4
withstand	b	0	3
withstood	n	1	2
withstood	d	1	1
witness	b	0	9
witnessed	d	0	7
witnessed	n	0	6
witnessing	g	0	6
wobble	b	0	3
wobbled	d	0	2
wonder	b	0	38

wondered	d	0	55
wondered	n	0	3
wondering	g	0	20
wonders	z	0	2
work	b	0	179
worked	d	0	76
worked	n	0	52
working	g	0	151
works	z	0	34
worry	b	0	43
worried	d	0	7
worried	n	0	28
worries	z	0	5
worrying	g	0	5
worship	b	0	5
worshiping	g	0	3
wound	b	0	2
wounded	n	0	22
wrangle	b	0	0
wrangled	d	0	2
wrap	b	0	5
wrapped	d	0	2
wrapped	n	0	12
wrapping	g	0	3
wreck	b	0	2
wrecked	d	0	2
wrecked	n	0	4
wrecking	g	0	5
wrench	b	0	0
wrenched	d	0	2
wrestle	b	0	2
wring	b	0	2
wrinkle	b	0	0
wrinkled	d	0	4
wrinkled	n	0	8
write	b	0	106
writes	z	0	41
writing	g	0	77
written	n	1	154
wrote	d	1	181
writhe	b	0	2
writhing	g	0	5
yank	b	0	1
yanked	d	0	4
yearn	b	0	1
yearned	d	0	2
yell	b	0	3
yelled	d	0	21
yelling	g	0	5
yield	b	0	16
yielded	d	0	7
yielded	n	0	5
yielding	g	0	8
yields	z	0	5

11.3 Appendix C. (The Ostrich test)

(51)

Your questions along with the child's answers.

1. **Q:** Where did the Ostreches live many year ago?

A: in Africa

2. **Q:** In Egypt who were allowed to wear the ostriche feathers and [illegible]

A: Kings, because they were important

3. **Q:** Why did their feathers endanger the birds?

A: Because The hunters killed them so people could ~~to~~ wear their feathers.

4. **Q:** Why were there not many Ostriches left in Africa

A: Because so many ostriches were killed for their [illegible] lay their eggs

5. **Q:** How do the farmer save the Ostriches, [illegible]

A: They take the eggs

Thank you very much for your time and help.

The information gathered by this questionnaire will be used for validating computer technology only. No personal names or details will be held with the responses. The information itself is provided on an entirely voluntary basis and will only be held at Bournemouth University.

(S2)

Your questions along with the child's answers.

1. Q: What kind of Birds were worshipped by the people of Egypt ?

A: Ostriches.

2. Q: Who were the only people allowed to wear their feathers?

A: Kings.

3. Q: Why were the ostriches nearly made extinct?

A: Hunters killed many of them for their feathers.

4. Q: How many eggs do they lay before they will sit on them?

A: Twelve.

5. Q: How do farmers get the birds to lay more eggs?

A: By hiding some of them.

Thank you very much for your time and help.

The information gathered by this questionnaire will be used for validating computer technology only. No personal names or details will be held with the responses. The information itself is provided on an entirely voluntary basis and will only be held at Bournemouth University.

Your questions along with the child's answers.

1. **Q:** Why did kings wear ostrich feathers?

   A: Because of their beauty.

2. **Q:** Why did hunters kill so many ostriches?

   A: To sell the feathers for people to wear

3. **Q:** What was the result of so many ostriches being killed?

   A: They nearly became extinct

4. **Q:** How many eggs do ostriches like to lay before sitting on them?

   A: 12 eggs

5. **Q:** How do ostrich farmers make the ostriches lay more eggs?

   A: By hiding some of the eggs

Thank you very much for your time and help.

The information gathered by this questionnaire will be used for validating computer technology only. No personal names or details will be held with the responses. The information itself is provided on an entirely voluntary basis and will only be held at Bournemouth University.

Your questions along with the child's answers.

1. **Q:** WHERE DO OSTRICHES COME FROM?

A: OSTRICHES COME FROM AFRICA.

2. **Q:** WHICH PEOPLE ONCE WORSHIPED OSTRICHES?

A: THE PEOPLE OF EGYPT ONCE WORSHIPED OSTRICHES.

3. **Q:** WHY WERE OSTRICHES HUNTED?

A: OSTRICHES WERE HUNTED FOR THEIR FEATHERS.

4. **Q:** HOW MANY EGGS DO OSTRICHES LAY?

A: OSTRICHES MAY ABOUT TWELVE EGGS.

5. **Q:** HOW MIGHT YOU 'FOOL' AN OSTRICH INTO LAYING EGGS.

A: YOU CAN 'FOOL' AN OSTRICH INTO HAVING MORE EGGS BY HIDING SOME THAT THEY HAVE ALREADY LAYED.

Thank you very much for your time and help.

The information gathered by this questionnaire will be used for validating computer technology only. No personal names or details will be held with the responses. The information itself is provided on an entirely voluntary basis and will only be held at Bournemouth University.

(55)

Your questions along with the child's answers.

1. **Q:** What did you used to hunt in Africa?

   A: Ostriches

2. **Q:** Why did people used to hunt ostriches?

   A: So they could sell or wear their feathers

3. **Q:** What was the result of killing so many ostriches?

   A: They nearly died out.

4. **Q:** What do ostriches lay?

   A: Eggs

5. **Q:** How do you make ostriches lay more eggs?

   A: Hide some of them

Thank you very much for your time and help.

The information gathered by this questionnaire will be used for validating computer technology only. No personal names or details will be held with the responses. The information itself is provided on an entirely voluntary basis and will only be held at Bournemouth University.

S6

Your questions along with the child's answers.

1. Q: How do farmers make Ostriches lay more eggs.

   A: Thiele some of them

2. Q: Can you remember the name of the animal in the story?

   A: Ostriches

3. Q: What country did Ostriches live in?

   A: Africa

4. Q: Why did hunter kill Ostriches?

   A: Rich people bought the feathers

5. Q: Where do Ostriches live today?

   A: Ostriches come from Africa.

Thank you very much for your time and help.

The information gathered by this questionnaire will be used for validating computer technology only. No personal names or details will be held with the responses. The information itself is provided on an entirely voluntary basis and will only be held at Bournemouth University.