An Adaptive E-Commerce Application using Web Framework Technology and Machine Learning

Georgia Isaac, Sofia Meacham, Hamed Hamzeh,
Angelos Stefanidis, Keith Phalp

Faculty of Science and Technology, Bournemouth University,
Fern Barrow, Poole, Dorset, BH12 5BB, UK
i7201048@bournemouth.ac.uk,
smeacham@bournemouth.ac.uk,hhamzeh@bournemouth.ac.uk,astefanidis@bournemouth.ac.uk, kphalp@bournemouth.ac.uk,

Abstract

In this paper, adaptivity and recommendation methods have been explored and implemented for an e-commerce web application of an online e-shop system, utilising Python web framework technologies.

The approach used to create such adaptivity methods is described through the analysis of initial requirements, models and designs of the planned solution, and the final implementation of the chosen method using the Web2py Python Model-View-Controller (MVC) framework. The formalisms used to achieve our goal, notably requirements documentation, Use Case diagrams for specification, and implementation, were investigated to determine their appropriateness for our case study. Two levels of solutions were provided: basic implementation using cookies functionality, and advanced implementation based on the integration of machine learning algorithms. As part of the advanced implementation, the suitability and advantages/disadvantages of different methods such as Scikit-learn, and general recommender systems, such as content-based recommendations, were analysed and presented. This type of implementation is the first step towards the explainable artificial intelligence (AI) paradigm where AI decisions are presented through adaptive interfaces. Lastly, future research possibilities are presented, by considering more applications and further design aspects.

Keywords: adaptive, web frameworks, recommender systems, Scikit-learn, explainable AI

1.0 Introduction

This paper discusses the integration of web applications with adaptive features developed using web framework technologies through the implementation of an e-commerce case study. Two levels of adaptivity are defined: basic, through the use of cookies, and advanced adaptivity through the integration of machine learning algorithms.

Adaptivity can be defined as an interactive software system which improves its ability to interact with a user, based on partial interactions with that user [1]. This improvement of interface interaction can be achieved utilising information stored in big data form, and processed through machine learning techniques, providing the user with more personalised recommendations.

Machine learning algorithms can be implemented alongside a user interface using various web technologies. It was established that sensible recommendations to users have already been given, using a system which was developed by leveraging PHP and SQL in 2005.The system used the Weighted Slope One algorithm to rank, and informedly, select items to recommend [2]. This method proved to be sufficiently relevant and usable, as recommendations are precomputable. However, in comparison to other collaborative filtering recommendation algorithms, both Improved Slope One and Weighted Slope One are outperformed [3].An improvement of this method could incorporate the PHP framework, Drupal. Drupal's built-in recommender API/module provides the developer with two recommendation options; "users who browsed this node also browsed", and "recommended for you" [4], an effective method for implementing machine learning algorithms with a user interface. It was observed, however, that the recommendations in this specific scenario were inaccurate; the implemented Drupal recommender was not enough and required a further content-based recommender [4].Python-based web frameworks such as Django have been utilised to create systems that use aspects of machine learning, namely feature extraction and classification, to generate item combinations for users [5]. Django has been used in collaboration with other web frameworks such as AngularJS to provide further improved interactions through personalisation for a user. A relevant study has demonstrated that, when using these frameworks in unison, more efficient personalisation results are achieved, provided they are appliedto a scenario [6].

Although there is literature suggesting that there is not extensive use of our chosen framework (Web2py), we were motivated by the strength of Python as a language and the ease-of-use of the framework itself. Web2py provides ease in rapid development, consisting of an in-built IDE and the simple Model-View-Controller (MVC) paradigm, supporting the work of both academic and scientific communities [7]. Based on past experience, the framework was simple to adopt, learn, and use. Alongside its ease of use, the framework's Python baselines enable the use of powerful scientific libraries that were explored when considering the implementation of machine learning. One such library is Scikit-learn, an open source machine learning library for Python, which supports the use of simple and efficient tools for data mining and data analysis [8]. With Web2py's simplicity and Scikit-learn's efficiency, it was agreed to leverage both technologies for implementation of a user interface with a machine learning algorithm.

Recommender systems are one example of the application of machine learning algorithms. Currently, these systems are more widely used where the application is low-risk, such as shop item recommendation, due to their unexplainable nature [9] and therefore lack of trustability.

The latest research on AI and its interfaces though, suggests that more is needed to establish trustability on AI decisions and interfaces leading to the Explainable AI paradigm. Explainable AI, is a paradigm described by DARPA as a capability that allows for the understandability, manageability, and essentially trustability of AI, required to resolve the non-intuitive, opaque, and incomprehensibility nature of machine learning [10]. With explainable AI, a chain of reasoning, based on the AI's knowledge and inference, can be provided to the user, demonstrating why the algorithm has made certain decisionsand not others[11].

Our proposed methodology is a first step towards developing adaptive interfaces designed appropriately to enable "confidence" in AI and enable the implementation of the future explainable AI paradigm.

The remainder of this paper will cover an overview of the case study in Section 2 to which the web application will be applied. In section 3, both high-level and low-level Use Case diagrams of the proposed web application are presented. Section 4 will detail implementation steps of both levels of adaptivity; cookies and machine learning-based. Section 5 will present reflections and evaluations of our implementation, and finally Section 6 offers conclusions and suggestions for future research directions.

2.0 Case Study Overview

This case study takes the form of a requirements document for an assignment set to students at Bournemouth University, studying on a Web Programming second year module. In this case study, requirements and suggested implementation methods were considered.

The resulting system will take the form of a large e-commerce website, the focussed section being a product review application.

The system will:

- Allow site administrators to view and search products that are being sold on the website.

- Allow site administrators to update product details, such as stock level, description, etc.

- Allow site administrators to add and delete products.

- Link to a back-end database where all product details must be stored.

- House a log-in system that authorises two user groups; general users, and administrative users.

- Display a list of products where logged-in users will be able to leave reviews for individual products.

- Allow logged-in users to view reviews by other users, as well as adding their own reviews.

In addition to the above requirements that define the basic system, we had to further enhance the system by implementing functionality resembling that of a recommender system. This was completed using basic functionality and would take the form of a 'Recommended Products' feature on the home page. Here, we expected products related to those recently viewed by the user, to be displayed, anticipating that the user would also be interested in those products, with a functionality that is similar to a content-based recommender system.

The technology of choice for such system was Web2py, a Python web framework which uses the Model-View-Controller (MVC) paradigm. Web frameworks are increasingly used in web development due to the abstraction they provide for common and reusable web development tasks enabling fast application development with substantially fewer lines of code.

3.0 System Design: Use Case Modelling

3.1 High-Level Use Case

To effectively analyse and understand the system's complete set of requirements, high-level Unified Modelling Language (UML) Use Case Modelling is applied, following the methodology outlined in [12]. These diagrams provide description for how a user/actor of the system should perceive the entire system, ensuring that all requirements previously stated, are met.

Fig. 1 shows a high-level Use case diagram, consisting of all scenarios mentioned in section 2.0, with the main actors being User, Administrative User, and System, and actions of Register, Log-in, Administrator Log-in, View Product (user), Leave Product Review (user), Update Product Details (admin), Add/delete Products (admin), and Recommend New Products (system).

In the case of recommendation of items, the interesting feature in this diagram is the System's action of Recommend New Products. This is further explored overleaf.

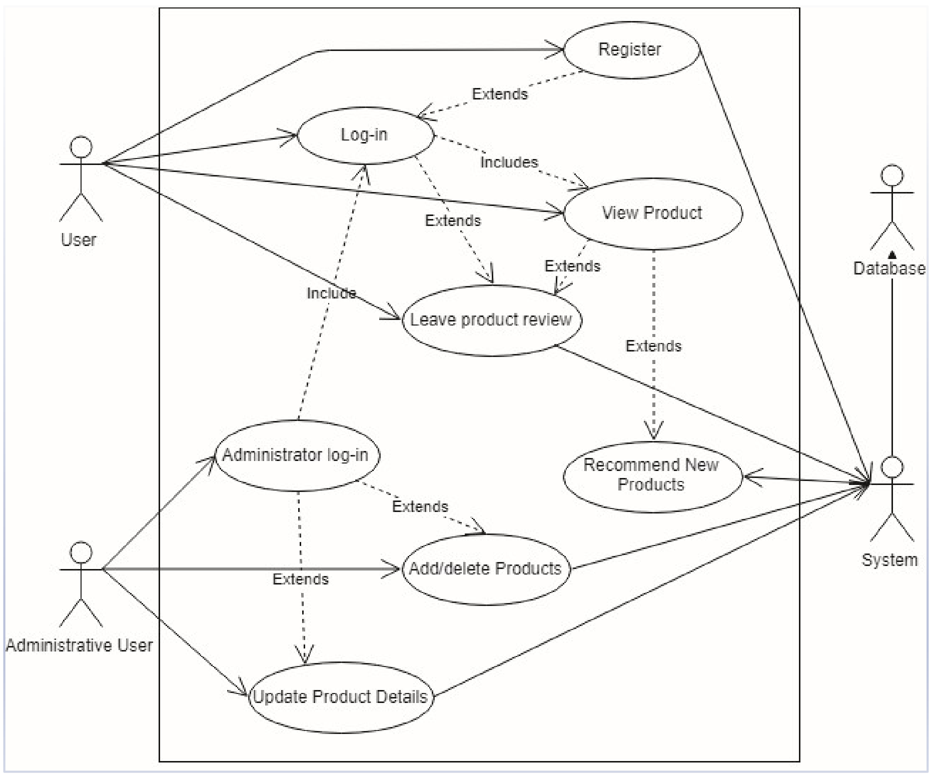


Figure 1. High-level UML Use Case Diagram of the system

3.2 Low-Level Use Case: Item Recommendation

To focus more on the item recommendation functionality, a lower-level Use Case was required to understand further actions needed to identify items recommend to users.

Fig. 2 shows a low-level Use Case diagram, capturing a more detailed definition of actions required by the system in order to recommend new items to a user. This process is irrespective of whether the user is logged in or not, as shown in Fig. 1.

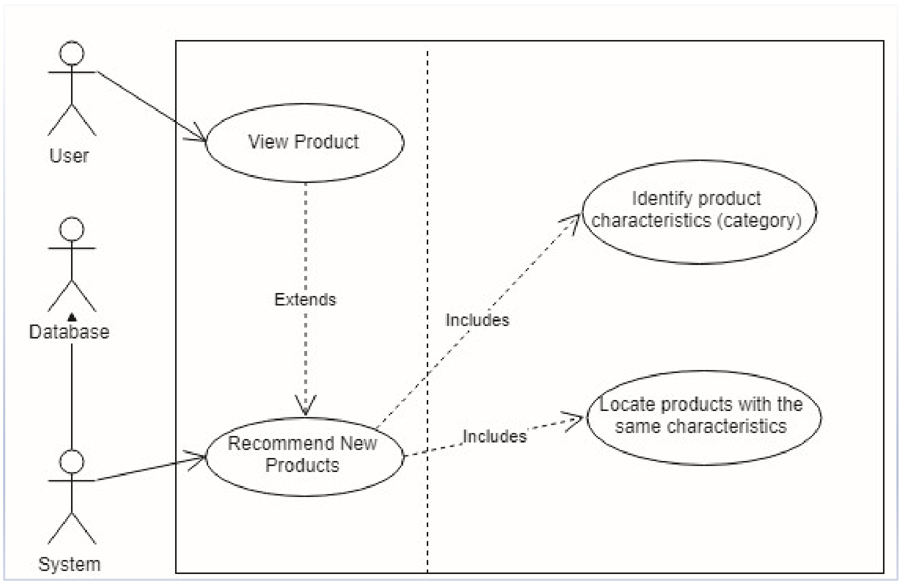

User
Database
System
View Product
Extends
Recommend New Products
Includes
Identify product characteristics (category)
Includes
Locate products with the same characteristics

Figure 2. Low-level UML Use Case Diagram for item recommendation

4.0 System Implementation

4.1 "Basic" Implementation using Cookies

In order to address the Use Case diagrams detailed above, a system was initially developed using the Web2py framework. This system forms both the main e-commerce website and the product review application, as specified in the requirements covered in section 2.0.

The system allowed for administrative users to manage (add, edit, and remove) products, and regular users to view, and leave reviews on products that are visible to other users.

The further development of the system to advance functionality, involved the development of the item recommendation feature to work alongside the initial system.As part of this development, the use of site cookies was our primary method.

Fig. 3 demonstrates that, firstly, cookies are created to hold information about the last product visited by a logged in user. *userID* corresponds to the unique identification number for a user, and *lastProd* stores the identification number of the product that was last viewed by the logged-in user.

```
def createLastProdCookies(userID, lastProd):
       response.cookies['user_id'] = userID
       response.cookies['user_id']['path'] = '/'
       response.cookies['last_prod_id'] = lastProd
       response.cookies['last_prod_id']['path'] = '/'
```

Figure 3. Controller – Creation of Cookies

Fig. 4 is the function call to create the cookies defined above, provided the user viewing the product page is currently logged-in. In the function call to *createLastProdCookies*, we observe that the two parameters match those required to set the cookies; the user's identification number (*auth.user_id*), and the viewed item's identification number (*post.id*).

```
if auth.is_logged_in():
       createLastProdCookies(auth.user_id, post.id)
```

Figure 4. Controller – Function used to create cookies if the user is logged in

Fig. 5 shows the check performed to identify whether, when a user is logged in and has visited a product's page, a cookie, containing the correct information, is set. If this check returns true, then the post variable is set to the last viewed product's unique identification number. Then, possible items to suggest are identified using a characteristic, namely 'category', of the last viewed product. The function proceeds to select three items from the products database, where their categories are the same as the category of the last viewed product. Using this logic, we are able to assume that, due to the user being interested in the initial product, they may also be interested in products from the same category which could be considered similar.

```
if request.cookies.has.key('user_id') and
request.cookies.has.key('last_prod_id'):
post = db.products(request.cookies['last_prod_id'].value)
suggestions =
       db.(db.products.category
       ==post.category).select(limitby=(0,3),
       orderby=~(db.products.id))
```

Figure 5. Controller –Function to build an array of suggested products

Once the controller has selected appropriate items to recommend, presently stored in the *suggestions* variable, Fig. 6 demonstrates the translation of these suggested products into the view for the user. Iterations of divider creation are completed for each suggested product, displaying product information such as name, and image. These products can now be viewed by the user.

```
{{for suggestion in suggestions}}
<div class=”SuggestedProduct”>
<div style=”width:80px;”>
<center>
     <a href=”{{=URL(‘product’, args =
     (suggestion.id,1))}}”><p>{{=suggestion.name}}</p>
     <img class=”ListedProductImage thumbnail”
     src=”{{=URL(‘download’,args=suggestion.image}}”/></a>
</center>
</div>
</div>
{{pass}}
```

Figure 6. View – HTML code to display the suggested products

4.2 “Advanced” Implementation using Machine Learning Algorithms

The previous section defineda simple cookie-based implementation for content-based recommendation. Implementation of the more advanced method with the Web2py interface would involve leveraging the Python Scikit-learn library, which is detailed in this section.

The recommendation of an item to a user, is based on a characteristic of the item,which is the item’s category. If the user has viewed an item in category 2, for instance, it is assumed that they will also like other items of the same category, therefore more items from category 2 are recommended to the user.

While this explanation is relatively simplistic, it demonstrates the limitations faced by providing recommendations using cookies. To provide a more accurate recommendation to a user, more characteristics of items should be considered, apart from the item’s category. To that end, a dataset containing each item and definition of the characteristics should be constructed. For instance, a popular application of recommender systems exists within the TV and movies domain, therefore a dataset for movies would require information such asthe movie name and its description. Figure 7 shows example movie data which is used in this example.

movie_id	Description
1	Quadruple trouble - action packed
2	The grandmother - crime thriller
3	Blue - romance
4	Insomnia - horror thriller dark
5	Funny animals - comedy funny
6	Sudden action - full of action and chase scenes
7	Camp funny - comedy funny
8	The sketchbook - romance comedy
9	Space cops - space action cop chase
10	ghosts - psychological dark

Figure 7. Fabricatedmovie 'items'

Building on the discussions thus far, the remainder of this section will focus on the implementation of content-based recommendation, using the dataset in Figure 7.

In order to use the item data with Scikit-learn, the pandas library is required to read in and manipulate the data. Figure 8 is an example of this in practice, using the movie dataset.

```
import pandas as pd
data = pd.read_csv('location\moviedata.csv')
```

Figure 8. Initial use of Pandas for reading the data shown in Figure 7

Following this, an algorithm to identify similarities based on the item's description should be used. One such algorithm is Term Frequency-Inverse Document Frequency (TF-IDF) used to identify words or characteristics with strong relationships to the item they belong to [13].

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

dataMatrix = data.as_matrix()
itemToCalculate = 8
similarItemsToShow = 3

tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 3),
min_df=0, stop_words='english')

tfidf_matrix = tf.fit_transform(data['description'])
```

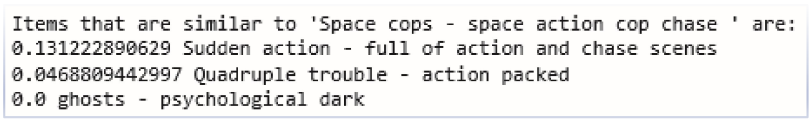
Figure 9. TF-IDF algorithm in Scikit-learn

Figure 9 displays the matrix of n-grams, which is the main feature of the TF-IDF algorithm. The *ngram_range* parameter defines three types of n-grams required to build the matrix: unigram (one word i.e. "action"), bigram (two words i.e. "action chase"), and trigram (three words i.e. "action chase scenes"). This matrix is designed to only contain words of relevance and ignore stop-words such as "the", "it", and "and", defined by the *TfidfVectorizer* parameter *stop_words='english'*. These words are not relevant to determining similarity between items and should therefore be ignored in n-gram creation. Regarding the hardcoded variables,*dataMatrix* allows easy access of the dataset, *itemToCalculate* defines which item in the dataset is the target item, and *similarItemsToShow* defines how many items we want to return that are similar to the target item. In this example, we want to find items that are similar to the movie "Space Cops".

```
Def find_similar(tfidf_matrix, index, top_n
=similarItemsToShow):

       cosine_similarities
       =linear_kernel(tfidf_matrix[index:index+1],
       tfidf_matrix).flatten()

       related_docs_indices = [I for I in
       cosine_similarities.argsort()[::-1] if I != index]

       return [(index, cosine_similarities[index]) for index
       in related_docs_indices][0:top_n]

message = ("Items that are similar to '%s' are: "
%dataMatrix[itemToCalculate][1])
print message

for index, score in find_similar(tfidf_matrix,
itemToCalculate):
        print score, dataMatrix[index][1]
```

Figure 10. Function for identifying similar items [14], and print statement for results

Figure 10demonstrates the function, *find_similar*, which will carry out the identification of similar items in the dataset, based on the target instance (*itemToCalculate*), based on a slight adaptation of Needham's TF-IDF implementation in Jupyter Notebook [14]. In simple terms, this function uses cosine similarity, a method of measuring the degree of similarity between a pair of text objects [15] where the most similar objects are parallel to each other, to identify the defined number of items (*similarItemsToShow*) that are similar to the target item. The similarity score and description of each similar item is then returned, displaying the results, as shown in figure 11.



```
Items that are similar to 'Space cops - space action cop chase ' are:
0.131222890629 Sudden action - full of action and chase scenes
0.0468809442997 Quadruple trouble - action packed
0.0 ghosts - psychological dark
```

Figure 11. Printed message displaying the results when requesting items similar to item 8 – Space Cops

Figure 11 demonstrates that, when given an item, the above Python code using Scikit-learn can identify similar items in order of similarity. The item ‘Sudden Action’ is deemed similar to ‘Space Cops’ as it is also described as an action movie, therefore is given a similarity score of 0.13. However, the movie ‘Ghosts’ is also provided (as we requested 3 similar items) which does not contain any similar characteristics to ‘Space Cops’ and is given a similarity score of 0.

Based on this, we conclude that the TF-IDF algorithm in Scikit-lean is effective in providing suitable recommendations for a content-based system, be an appropriate implementation approach which would improve section 4.1.

5.0 Reflections and evaluationof the approach

In this paper, two implementations for content-based item recommendation were demonstrated.

Our initial use of cookies formed an effective method of item recommendation in the context of a small e-commerce store for university assignment purposes. It enabled us to provide the user with a list of items that they may be interested in, based on an item that they had previously used.

Although this method of recommendation is seen as useful to some extent [16], a more comprehensive application of this method could face issues concerning cookie churn, where the amount of data stored in cookies becomes too arduous to work through using cookies alone. In this situation, Yahoo! suggests the use of machine learning algorithms to overcome the issue [17].

As a result, we presented a new implementation approach, utilising the same web framework (Web2py) along with a Python library for machine learning (Scikit-learn). This implementation utilised the use of the TF-IDF algorithm, available in Scikit-learn, which performs best in situations where relationships of items based on keywords must be found [18].

In contrast, the code required for the cookie-based implementation required significant fragmentation, with sections of code appearing in multiple separate sections of the application’s controller in Web2py. The machine learning implementation required a small amount of < 20 lines of code, which can be located in the same area of the controller, as the main workings of the code consists of one function call to *find_similar()*.

When using machine learning for tasks such as item recommendation, the problem of AI trustability is introduced. When presented with a given result, i.e., the movie “Sudden Action” is similar to “Space Cops”, one may ask the question of “why?”. In situations where a critical decision is being made by a machine learning algorithm, such as one which may affect a person’s wellbeing, we may not want to trust an algorithm that does not give valid reasoning for its decision to avoid repercussions, if the decision is deemed erroneous.

In this instance, it could be possible to explain the decisions made by the machine learning algorithm by returning information such as the specific keywords that were found when comparing items to each other. Furthermore, data used to determine cosine similarity in the TF-IDF algorithm could be extracted and translated for a user to understand. Perhaps this would give more insight into why items are identified as being similar and therefore improve the trustability of this implementation’s algorithm.

6.0 Conclusions and future work

This paper has presented the integration of web applications with adaptive features using the Web2py web framework and the Scikit-learn Python library, using a university assignment case study. Our first method used web cookie technology to provide content-based item recommendations to users of an e-commerce web system. Our second method improved on the previous methodand discussedthe implementation of a machine learning algorithm to provide content-based item recommendations, where the use of cookies may not be sufficient for large scale applications.

Our future research plans focus on the application of machine learning algorithms to more applications that use web framework technology, with specific emphasis on improving the trustability of said machine learning algorithms. Effective implementation of explainability for machine learning algorithms will be explored and implemented too. Lastly, this further implementation will demonstrate the ability to develop adaptive web interfaces using web frameworks, controlled by “well-explained” machine learning algorithms.

7.0 References

1. Langley, P.,1997. *Machine Learning for Adaptive User Interfaces*. In: KI '97 Proceedings of the 21st Annual German Conference on Artificial Intelligence: Advances in Artificial Intelligence. [online] Available at: https://dl.acm.org/citation.cfm?id=731753 [Accessed 6 Feb. 2018].

2. Lemire, D. and McGrath, S., 2005. *Implementing a Rating-Based Item-to-Item Recommender System in PHP/SQL*. Technical Report D-01. [online] Available at: https://www.researchgate.net/profile/Daniel_Lemire/publication/239218189_Implementing_a_Rating-Based_Item-to-Item_Recommender_System_in_PHPSQL/links/53e949db0cf2dc24b3cab31b.pdf [Accessed 6 Feb. 2018].
3. Wang, P., Qian, Q., Shang, Z. and Li, J., 2016. *An recommendation algorithm based on weighted Slope one algorithm and user-based collaborative filtering*. In: Control and Decision Conference (CCDC), 2016 Chinese. [online] IEEE. Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7531393 [Accessed 6 Feb. 2018].
4. Wang, P. and Yang, H., 2012. *Using collaborative filtering to support college students' use of online forum for English learning*. Computers & Education, [online] 59(2), pp.628-637. Available at: https://ac.els-cdn.com/S0360131512000577/1-s2.0-S0360131512000577-main.pdf?_tid=6f4aef02-0b73-11e8-89f4-00000aacb361&acdnat=1517945285_1dcc56bc43e1f3f6d287185d410a3cb6 [Accessed 6 Feb. 2018].
5. Vartak, M. and Madden, S., 2013. *CHIC: A Combination-based Recommendation System*. Proceedings of the 2013 internationalconference on Management of data - SIGMOD '13. [online] Available at: https://people.csail.mit.edu/mvartak/papers/chic.pdf [Accessed 6 Feb. 2018].
6. Vidaković, D., Segedinac, M., Obradović, Đ. and Savić, G. (2017). *A Recommendation System with Personalizable Distributed Collaborative Filtering*. In: 7th International Conference on Information Society and Technology ICIST 2017. [online] Eventiotic. Available at: http://www.eventiotic.com/eventiotic/files/Papers/URL/be5cd6e3-2798-4a9b-a93f-4ae1717bbd28.pdf [Accessed 6 Feb. 2018].
7. Di Pierro, M. (2011). *web2py for Scientific Applications*. Computing in Science & Engineering, [online] 13(2), pp.64-69. Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5518770 [Accessed 7 Feb. 2018].
8. *Scikit-learn: Machine Learning in Python*, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
9. Mcsherry, D., (2005). *Explanation in Recommender Systems*. Artificial Intelligence Review, 24 (2), 179-197. [Accessed 22 Feb. 2018]
10. Gunning, D., (2016). Explainable Artificial Intelligence (XAI). Arlington, VA: DARPA.
11. Lent, M., Fisher, W., Mancuso, M. (2004). *An Explainable Artificial Intelligence System for Small-unit Tactical Behavior.* Proceedings of the 2004 conference on Innovative applications of artificial intelligence. [online] Available at: https://www.aaai.org/Papers/IAAI/2004/IAAI04-019.pdf [Accessed 22 Feb. 2018]

12. Meacham, S. and Phalp, K. (2016). *Requirements engineering methods for an Internet of Things application: fall-detection for ambient assisted living.*In: BCS SQM/Inspire Conference. [online] ResearchGate. Available at: https://www.researchgate.net/publication/309385353_Requirements_engineering_methods_for_an_Internet_of_Things_application_fall-detection_for_ambient_assisted_living [Accessed 25 Feb. 2018].
13. Ramos, J., 2003. *Using TF-IDF to Determine Word Relevance in Document Queries*. In: The First instructional Conference on Machine Learning (iCML-2003) [online]. Piscataway: Rutgers. Available from: https://www.cs.rutgers.edu/~mlittman/courses/ml03/iCML03/papers/ramos.pdf [Accessed 23 Feb 2018].
14. Needham, M., 2016. *scikit-learn: TF/IDF and cosine similarity for computer science papers* [online]. markneedham.com. Available from: http://www.markhneedham.com/blog/2016/07/27/scitkit-learn-tfidf-and-cosine-similarity-for-computer-science-papers/ [Accessed 23 Feb 2018].
15. Thada, V. and Jaglan, V., 2013. *Comparison of Jaccard, Dice, Cosine Similarity Coefficient To Find Best Fitness Value for Web Retrieved Documents Using Genetic Algorithm*. International Journal of Innovations in Engineering and Technology (IJIET) [online], 2 (4). Available from: https://pdfs.semanticscholar.org/8575/e8beef47bd2880c92f54a749f933db983e56.pdf [Accessed 23 Feb 2018].
16. IBM (2014). *Product Recommendations Cookies*. [online] IBM Knowledge Center. Available at: https://www.ibm.com/support/knowledgecenter/en/SSPJVK/DigitalRecommendations/UserGuide/intel_cookies.html [Accessed 24 Feb. 2018].
17. Dasgupta, A., Gurevich, M., Zhang, L., Tseng, B. and Thomas, A. (2012). *Overcoming browser cookie churn with clustering*. In: Fifth ACM international conference on Web search and data mining. New York, NY: ACM.
18. Kazemi, B. and Abhari, A. (2017). *A comparative study on content-based paper-to-paper recommendation approaches in scientific literature*. In: Communications & Networking Symposium. ACM.