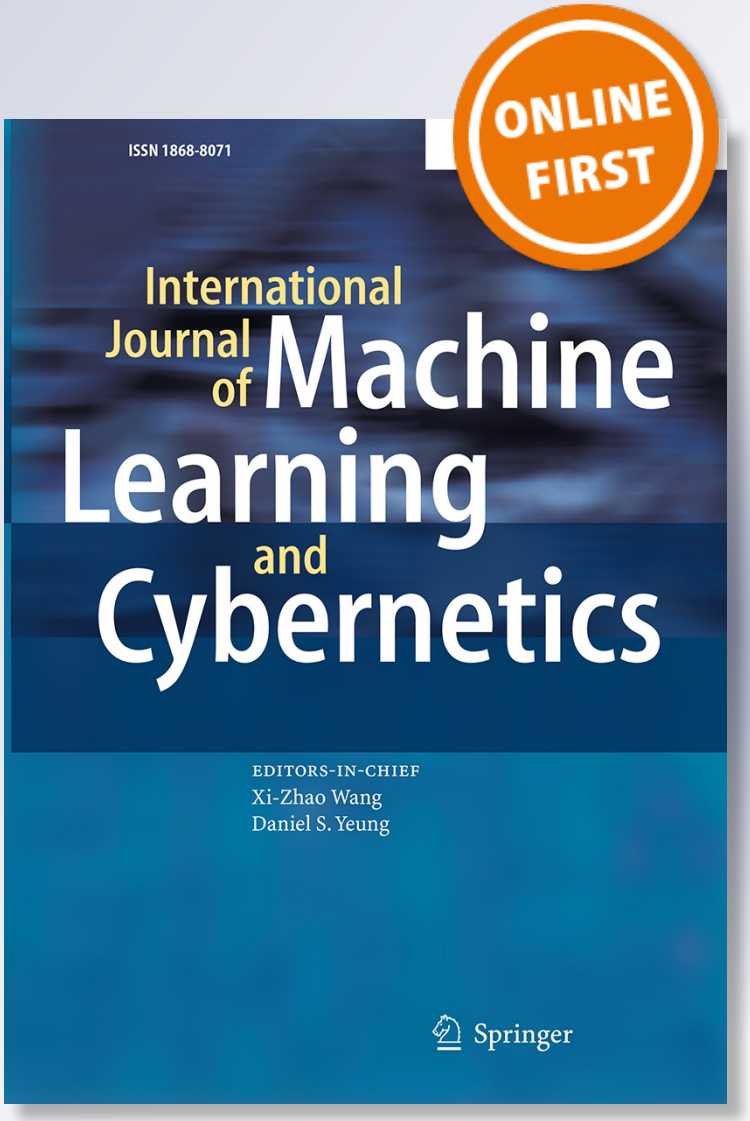
Improving news articles recommendations via user clustering

Christos Bouras & Vassilis Tsogkas

International Journal of Machine Learning and Cybernetics

ISSN 1868-8071

Int. J. Mach. Learn. & Cyber.
DOI 10.1007/s13042-014-0316-3



Springer

Int. J. Mach. Learn. & Cyber.
DOI 10.1007/s13042-014-0316-3

ORIGINAL ARTICLE

Improving news articles recommendations via user clustering

Christos Bouras · Vassilis Tsogkas

Received: 2 September 2014 / Accepted: 18 November 2014

Abstract Although commonly only item clustering is suggested by Web mining techniques for news articles recommendation systems, one of the various tasks of personalized recommendation is categorization of Web users. With the rapid explosion of online news articles, predicting user-browsing behavior using collaborative filtering (CF) techniques has gained much attention in the web personalization area. However common CF techniques suffer from problems like low accuracy and performance. This research proposes a new personalized recommendation approach that integrates both user and text clustering based on our developed algorithm, W-kmeans, with other information retrieval (IR) techniques, like text categorization and summarization in order to provide users with the articles that match their profiles. Our system can easily adapt over time to divertive user preferences. Furthermore, experimental results show that by aggregating item and user clustering with multiple IR techniques like categorization and summarization, our recommender generates results that outperform the cases where each or both of them are used, but clustering is not applied.

Keywords News clustering · k-means · W-kmeans · Cluster labeling · Partitional clustering · Collaborative filtering

1 Introduction

Relying on recommendations from other people is a basic means of filtering though the vast amount of information that the average internet user comes across every day. Recommendations can be in the form of spoken words, reference letters, reports from news media, general surveys, travel guides, site reviews, etc. This natural social process is assisted by recommender systems that have risen over the last 15 years at many large electronic sites and which aim to help people shift through available news articles, web pages, movies, and so on, in order to find the most interesting and valuable piece of information for them.

Recommendations can roughly be divided into the following approaches: (a) content-based, where users are profiled by identifying their characteristic features—something that requires personal data which are difficult to harvest and (b) collaborative filtering (CF), where we take advantage of the fact that people who had similar tastes in the past may also agree on their tastes in the future.

Collaborative filtering techniques use a database of preferences for items by users to predict additional topics or products a new user might like. In order to conceptualize the process that is usually followed by CF systems in general, consider the list of viewed articles per user presented in Table 1.

This manuscript is an extended version of the paper by Christos Bouras and Vassilis Tsogkas entitled: "User Personalization via W-kmeans." Frontiers in artificial intelligence and applications, volume 243: advances in knowledge-based and intelligent information and engineering systems.

C. Bouras (✉) · V. Tsogkas
Computer Engineering and Informatics Department, University of Patras, Patras, Greece
e-mail: bouras@cti.gr

V. Tsogkas
e-mail: tsogkas@ceid.upatras.gr

C. Bouras
Computer Technology Institute and Press "Diophantus", 26500 Rio Patras, Greece

Published online: 30 November 2014

Table 1 List of articles each user has expressed interest to

User	Article #
Bill	Article 1, Article 2
Anti	Article 1, Article 2, Article 3
Helen	Article 2, Article 4
Alex	Article 2, Article 4, Article 5
George	Article 1

First of all, we observe that there is overlapping article interest between users, a situation commonly true on a real user system and which is exploited within the scope of any CF technique. Say the user 'George' returns, Article 2 is not useful for recommendation since everyone has read it. On the other hand, Article 3 might be a good suggestion for him.

Object clustering refers to the process of partitioning a collection of objects into several sub-collections based on their similarity of contents. For the case of user clustering, each sub-collection is called a user cluster and includes users that have revealed similar appeals in their selections of text articles while browsing through a document collection. Clustering has been proven to be a useful technique for information retrieval by discovering interesting information kernels and distributions in the underlying data. In general, it helps constructing meaningful partitions of large sets of objects based on various methodologies and heuristics thus playing a crucial role in organizing large collections. Clustering has been used: (a) to structure query results, (b) in order to assist detection of the organized topical groups within the document, in tandem with other information retrieval techniques–such as text summarization, (c) to improve the performance of recommendation systems by affecting the suggestions made towards the end users. Despite the obvious benefits of exploiting clustering, there are not many CF systems nowadays that take it consideration for generating personalized content.

From the beginning of the Web and the Internet in general, the usual means of accessing information concerning news that occur around the world, has been Internet news portals. The amount and diversity that news portal, as well as the news content that they serve, has grown exponentially ever since. With the arrival of Web 2.0 though, new protocols and methods both for formatting and presenting information have changed the status quo. Users are no mere consumers but, participate actively to the information production procedure, providing feedback via a variety of means. Moreover, they are regularly frustrated by the poor filtering capabilities that most news portals provide, since many tools are suitable for searching and not filtering information.

Motivated by above issues, in this manuscript, we describe a novel recommendation engine that is utilizing a combination of various techniques and heuristics originating from the information retrieval (IR) domain, and most notably, user and item clustering, in order to generate news articles suggestions for the system users that match their continuously updating profiles and thus improve the *user experience* of our recommender. By the term *user experience* we describe the delivery of the recommendations to the user and the interaction of the user with those recommendations.

The rest of the manuscript is structured as follows. In Sect. 2 an outline of the recent works concerning Web mining, clustering, CF and recommendation techniques is given. In Sect. 3 we present the flow of information that is followed within our mechanism starting from the news articles' fetching step and resulting to the personalized recommendations. In Sect. 4 we describe the algorithms concerning user clustering and personalization that are used in our approach. In Sect. 5 the experimental procedure and its results are presented. Section 6 gives some concluding remarks regarding this work while Sect. 7 briefly outlines some future enhancements.

2 Related work

As explained by Konstan and Riedl [14], recommender systems have a history spanning from a focus on prediction algorithms that was later extended to the commercial world and is currently focusing on more elaborate methodologies that are moving beyond the accurate predictions. A good analysis of how recommender systems spring up and evolved over time is given in the extensive work of Ekstrand et al. [8].

Web mining focuses on finding natural groupings of Web resources or Web users. We could roughly divide Web Mining into three basic categories [6]. First, Web content mining, where information is extracted from the content of pages and links (i.e. not from the users themselves). Second, Web Structure Mining, where structural information about hyperlinks and organization plays a predominant role. And third, Web Usage Mining which focuses on extracting useful usage patterns from the users' behavior. Clustering of Web users is a particular research topic of Web Usage Mining that aims towards describing generic trends in users' behaviors within some particular time (e.g. a specific time-window). As explained by Pavlov, and Pennock [28], Web Mining is the extraction of interesting and potentially useful patterns and implicit information from artifacts or activity related to the World Wide Web. The field has also been explored within the scope of Web personalization by various works, e.g.

Eirinaki, and M. Vazirgiannis [7] and Fu et al. [9]. Mobasher et al. [25] take into account basically two types of usage patterns and cluster them in order to build generic navigational profiles, without minding the order of accesses. A similar approach was also recently used by Li et al. [18] where an approach to construct the approximate concept lattice of an incomplete context is described. A method that uses attribute-oriented induction where user sessions are represented as vectors in an n-dimensional Euclidian term space is described by Fu et al. [9]. A visualization approach of the user choices has also been explored by Cadez et al. [4] for navigation patterns. Hay et al. [11] introduce a Sequence Alignment Methodology that clusters users based on their navigation patterns. This work focuses on the order in which navigation events take place by users.

Web usage mining results to CF when it uses the known preferences of a group of users to make recommendations or predictions about the unknown preferences for other users.

2.1 Collaborative filtering

Collaborative filtering was initially introduced in order to describe the previously mentioned personalized recommendation technique. Ever since, CF has been widely adopted and has evolved so that recommenders may suggest particularly interesting items, in addition to indicating those that should be filtered out. The fundamental assumption of CF [16] is that if users X and Y rate n items similarly, or have similar behaviors (e.g. buying, watching, listening), they will rate or act on other items similarly. Several matric factorization techniques have been applied to CF, like SVD, probabilistic LSA, probabilistic matrix factorization, etc. However, the combination of various algorithms appears to outperform single methodologies [31]. CF techniques use a database of preferences for items by users to predict additional topics or products a new user might like. In a typical CF scenario, there is a list of m users $\{u_1, u2,\ldots, um\}$ and a list of n items $\{i_1, i_2,\ldots, i_n\}$, and each user, u_i, has a list of items, I_{ui}, which the user has rated, or about which their preferences have been inferred through their behaviors. The ratings can either be explicit indications, on a 1–5 scale, or implicit indications, such as purchases or click-throughs. CF algorithms are required to have the ability to deal with highly sparse data, to scale with the increasing numbers of users and items, to make satisfactory recommendations in a short time period, and to deal with other problems like synonymity (the tendency of the same or similar items to have different names), shilling attacks, data noise, and privacy protection problems [31].

CF techniques come in roughly three categories: (a) memory based, like neighbor-based and item-based top-N, (b) model-based, like Bayesian belief nets, latent semantic, dimensionality reduction (SVD) and (c) hybrid, which combine the advantages of both categories and improve the prediction performance. Early generation collaborative filtering systems, used the user rating data to calculate the similarity or weight between users or items and make predictions or recommendations according to those calculated similarity values. Memory-based CF methods are notably deployed into commercial systems such as Amazon (amazon.com) and Barnes and Noble [12], because they are easy-to-implement and highly effective. Customization of CF systems for each user decreases the search effort for users.

One basic problem with CF is that it doesn't always work well due to data scarcity, also known as 'new user problem'. Each person has seen only a small fraction of the data, thus accurate predictions cannot be easily made until the coverage of users/data has increased to a significant value. One way to deal with this situation is to group people into clusters of similar interests. This way, by using symmetry, one might group articles based on who sees them and use article groups to users. For example, consider a group of articles that might be about politics and Obama in particular. A user that has previously showed some interest towards reading Obama or democrat-related articles might also want to view articles from this group. A reverse approach is also possible: consider a group of users that have previously expressed their interest for this kind of topic. A newly added article with similarities to some of the articles previously read by the people of this group might also be appealing to the rest of the group. The above approach suggests that instead of depending on choices of single users, the cluster aggregates the needed information. Two widely used techniques for this scenario has traditionally been k-NN and clustering (usually k-means or a variance), or a combination of both [26], Li and Chung[19] Maier et al. [22]. In this manuscript, we derive those groups or classes by arranging the information that is extracted from a variety of IR techniques, like categorization, clustering and also inferred by previous user behavior. Another approach that has been successfully used to tackle with the new user problem is matrix factorization [15].

Another problem of CF is that similarity scores typically do not take into consideration the user interest shifting. Also, they do not estimate the reliability of the user choices which can lead to poor recommendation results. We are trying to tackle this problem by rather small but continuous user profile adjustments.

To achieve better prediction performance and overcome shortcomings of memory-based CF algorithms, model-based CF approaches have been investigated. Model-based CF techniques use the pure rating data to estimate or learn

a model to make predictions. The model can be a data mining or machine learning algorithm. Well-known model-based CF techniques include Bayesian belief nets (BNs) CF models [24] and Shani et al. [30], clustering CF models [32] and Chee et al. [5], and latent semantic CF models [12]. An MDP (Markov decision process) based CF system [29] produces much higher performance than a system that has not deployed the recommender.

Besides collaborative filtering, content-based filtering is another important class of recommender systems. Content-based recommender systems make recommendations by analyzing the content of textual information and finding regularities in the content, for example Hannon et al. [10]. The major difference between CF and content-based recommender systems is that CF systems only use the user-item ratings data to make predictions and recommendations, while content-based recommender systems rely on the features of users and items for predictions Shani et al. [30]. Both content-based recommender systems and CF systems have limitations. While CF systems do not explicitly incorporate feature information, content-based systems do not necessarily incorporate the information in preference similarity across individuals [1]. Hybrid CF techniques, such as the content-boosted CF algorithm [23] and Personality Diagnosis (PD) [27], combine CF and content-based techniques, hoping to avoid the limitations of either approach and thereby improve recommendation performance.

2.2 Accompanying personalization with clustering

Personalized search is an important research area that aims to resolve the ambiguity of query terms. To increase the relevance of search results, personalized search engines create user profiles to capture the users' personal preferences and as such, identify the actual goal of the input query. Since users are usually reluctant to explicitly provide their preferences due to the extra manual effort involved, recent research has focused on the automatic learning of user preferences from users' search histories or browsed documents and the development of personalized systems based on the learned user preferences. Most approaches have employed a single large user profile for each user in the personalization process. In reality, positive preferences are not enough to capture the fine-grain interests of a user. User profiling strategies can be broadly classified into two main approaches: document-based and concept-based approaches. Document-based user profiling methods aim at capturing users' clicking and browsing behaviors. Users' document preferences are first extracted from the click-through data and then used to learn the user behavior model which is usually represented as a set of weighted features. On the other hand, concept-based user profiling methods aim at capturing users' conceptual needs. Users' browsed documents and search histories are automatically mapped into a set of topical categories. User profiles are created based on the users' preferences on the extracted topical categories. Joachims [13] discusses of a method which employs preference mining and machine learning to model users' clicking and browsing behavior was proposed. This method assumes that a user would scan the search result list from top to bottom. If a user has skipped a document d_i at rank i before clicking on document d_j at rank j, it is assumed that he/she must have scan the document d_i and decided to skip it. Thus, we can conclude that the user prefers document d_j more than document d_i (i.e. $d_j < r\ d_i$, where r is the user's preference order of the documents in the search result list).

Varelas et al. [34] focus on the personalized recommendation of Web pages that are adapted according to the access patterns constructed by analyzing user navigation information. They prove that the methodology of integrating user clustering within the scope of a recommendation system, while mining interesting user navigation patterns can be beneficial. Lops et al. [20] predict the preferences of a user for an item by weighting the contributions of similar users, called neighbors, for that item. Similarity between users is computed by comparing their rating styles, i.e. the set of ratings given on the same items or by means of their browsing habits. Moreover, clustering of partial preference relations such as a means for agent predictions for user preferences has been investigated in Qin et al. [27]. In addition to the above, the chosen similarity measure regarding the clustering approach is of great significance. For example in Yeung and Wang [36] the authors shows that, by using a gradient descent technique to learn the feature weights, the clustering performance can be significantly improved.

In our previous work [2], we proposed a new clustering methodology which extends the original W-kmeans algorithm by enriching the results with WordNet hypernyms. WordNet is one of the most widely used thesauri for English. It attempts to model the lexical knowledge of a native English speaker. Containing over 150,000 terms, it groups nouns, verbs, adjectives and adverbs into sets of synonyms called synsets. The synsets are organized into senses, giving thus the synonyms of each word, and also into hyponym/hypernym (i.e. Is-A), and meronym/holonym (i.e. Part-Of) relationships, providing a hierarchical tree-like structure for each term. The applications of WordNet to various IR techniques have been widely researched concerning finding the semantic similarity of retrieved terms [33], or their association with clustering techniques. Moreover, as opposed to our approach (utilizing WordNet hypernyms), synsets for word-sense

disambiguation have also been used with success by Lops et al. [20].

In this manuscript we present the application of this algorithm within a more generic framework. Our goal is to improve the results of our information retrieval system in terms of precision/recall, and thus serve better filtered and adequate results to their users, helping in essence the decision making process. Since we are dealing with the effective and adequate retrieval of personalized news articles that are retrieved from the web, we present the personalization algorithm that is used for presenting the categorized, clustered and summarized articles to the user. Our recommendation approach can be classified as 'hybrid' since it is mainly content-based with some collaborative filtering features that enhance the algorithm with the ability to automatically adopt over time to the continuously changing user choices. Furthermore, we base our summarization procedure (enhanced by the categorization module) on the TF-IDF term-weighting model. Our recommendation engine incorporates several heuristics such as the viewed articles by the user, the time a user spends on reading an article, the categorization of the articles, the clustering of articles and registered system users. In essence, we are able to decode the navigation patterns of users and aggregate their profiles using the W-kmeans clustering algorithm. This allows our recommendation system to suggest content that, with high probability, will be interesting to the users.

3 Information flow

Figure 1 depicts the flow of information within our recommendation system. Initially, at its input stage, news articles are crawled and fetched from news portals from around the Web. This is an offline procedure and once articles as well as metadata information are fetched, they are stored in the centralized database from where they are picked up by the procedures that follow. More information regarding the afore-mentioned processes can be found by Bouras et al. [3].

A key process of the system as a whole, probably as important as the clustering algorithm that follows it, is text preprocessing on the fetched article's content, that results to the extraction of the keywords each article consists of. Analyzed in Bouras et al. [3], keyword extraction handles the cleaning of articles, the extraction of the nouns, the stemming as well as the stopword removal process. Following, it applies several heuristics to come up with a weighting scheme that appropriately weights the keywords of each article based on information about the rest of the documents in our database. Pruning of words, appearing with low frequency throughout the corpus, which are unlikely to appear in more than a small number of articles, comes next. Keyword extraction, utilizing the vector space model, generates the term-frequency vector, describing each article as a 'bag of words' (words—frequencies) to the key information retrieval techniques that follow: article categorization, summarization and clustering.

The main scope of the categorization module is to assist the summarization procedure by pre-labeling the article with a category and has been proven to provide better results [3]. What's more, they interact with each other in order to improve the results when each of them is used separately. Summarization then proceeds with extracting a short but useful piece of textual information that can convey the article's meaning. Following those two core processes, our recommendation approach extracts session data based on the paths that are recorded for each registered system user.

As far as our clustering approach is concerned, our aim towards increasing the efficiency of the used clustering algorithm is to enhance the aforementioned 'bag of words' with the use of an external database, WordNet. The above characteristics of our system give its content-based nature. This enhanced feature list, 'feeds' the k-means clustering procedure that comes next and is depicted in Fig. 2. Note however that user and item clustering are two separate procedures (as shown in Fig. 1), but are still using the same kernel process to generate their clusters as depicted in Fig. 2.

In the current work, clustering is achieved via a variation of the regular k-means algorithm using the cosine similarity distance measure:

$$d(a,b) = \cos(\theta) = \frac{a \times b}{|a||b|} \quad (1)$$

where |α|, |b| are the lengths of the vectors α, b, respectively and the similarity between the two data points is viewed by means of their angle in the n-dimensional space. It is important to note, however, that the clustering process is independent of the rest of the steps, meaning that it can easily be replaced by any other clustering approach operating on a word-level of the input documents.

Following the core IR tasks of our mechanism, the personalization algorithm takes place. The personalization module (dashed in Fig. 1) that is described in this manuscript, is easily adaptable to the user, meaning that small changes to the user's preferences, as expressed by his/her browsing behavior, are detected, thus adjusting continuously the user's profile. The algorithm uses a variety of user-related information in order to filter the results presented to the user. Furthermore it takes into account, in a weighted manner, the information originating from the previous levels regarding the summarization/categorization and news/user clustering steps. For each user viewing news

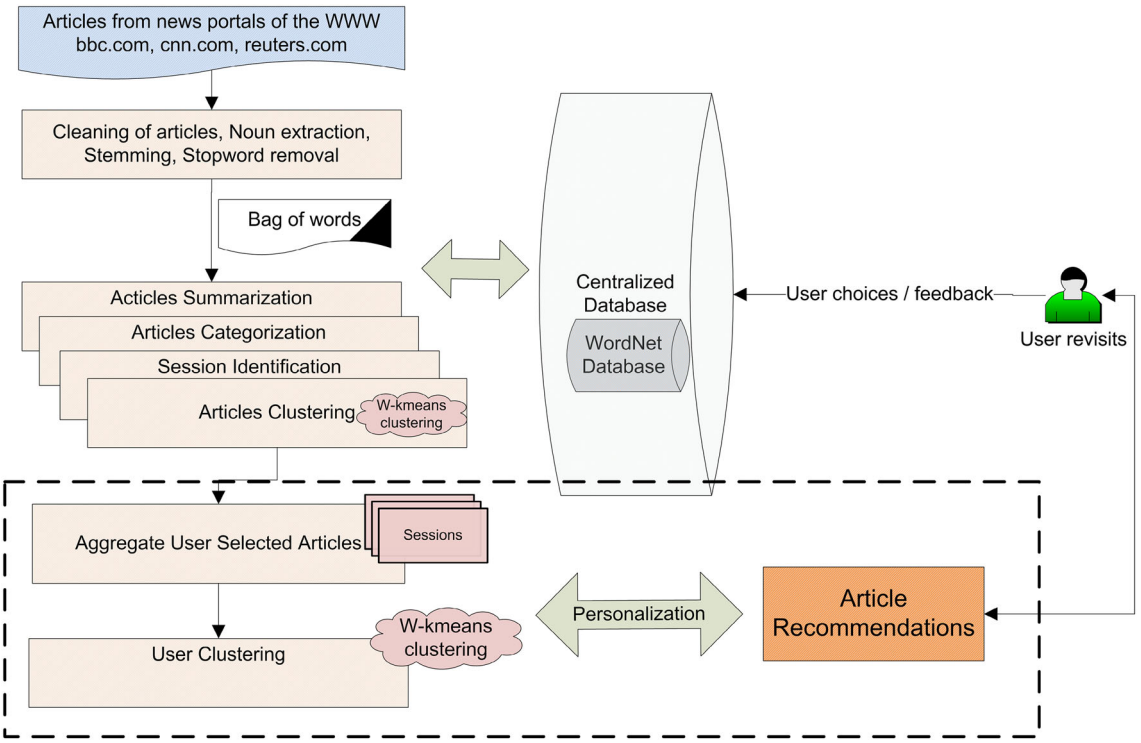


Fig. 1 Flow of information

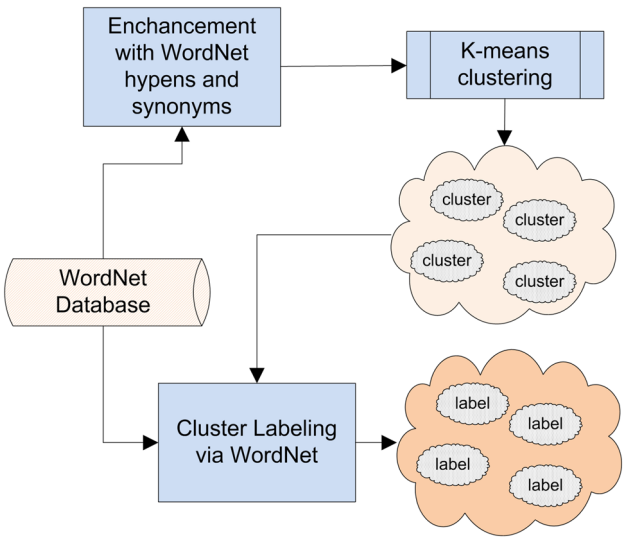


Fig. 2 Article/user clustering process

articles, we keep track of the selected actions which characterize a user session.

For connecting the user clustering component with our personalization algorithm, we define the notion of a session as the list of selected articles that a user has decided to view for a minimum duration and within a limited time frame, both of which are fine-tuned at the experimentation stage. The selected articles contained in those sessions are then aggregated at a keyword level generating a time-limited user profile. User profiles from multiple users and timeframes are then clustered using the W-kmeans [2] algorithm forming profile clusters.

W-kmeans is a novel approach that extends the standard k-means algorithm by using the external knowledge from WordNet hypernyms for enriching the "bag of words" used prior to the clustering process. The W-kmeans algorithm enhances the user profiles with hypernyms deducted from the WordNet database, using a heuristic manner (explained later in this manuscript). Those profile clusters, being essentially user clusters, are used at the recommendation stage to enhance the system's usage experience by providing better adapted results to users revisiting the site. Following the session clustering procedure, the resulting clusters are labeled using our WordNet cluster labeling mechanism.

When a user comes back, his clustered profile is recalled. Articles matching his profile are extracted and are considered for user recommendations. Suggested articles do not belong to the ones the user has already visited and also are not closely related to articles that the user has marked negatively in the past.

The approach previously described is essentially the collaborative filtering nature of our recommendation system, which practically involves related users to the decision making process. We expect that combining this

method with our keyword extraction (content-based) mechanism, the recommendations towards users will ameliorate.

4 Algorithm approach for recommendations via W-kmeans

The proposed approach consists of three major algorithmic components that are used for: (a) the offline process of identifying the sessions of users navigating through the recommendation system, (b) the offline process of clustering the detected sessions, and (c) the online process of recommending news articles to the users based on the clustered profiles. Those components are: session identification, clustering of user sessions and recommendation stage.

4.1 Session identification

The identification of sessions within a user's browsing history is achieved using Algorithm 1. This algorithm uses two important threshold values: (a) the viewing threshold, i.e. the minimum amount of time that a user is expected to spend in an article that is of interest to him/her, and (b) the session threshold, i.e. the maximum amount of time that on average a user is expected to spend continuously at our service while viewing articles.

Those two values are key with regards to determining when a user session begins and where it ends. On one hand, we do not want to miss out a continuous article selections (also known as click-streams in recommendation systems). On the other hand however, we should not wait too much so as for the session to include several user tastes as generally expressed by a larger waiting window. In order to tune correctly the values for (a) and (b), we analyzed the browsing patterns of the users as they are already registered into our database. In particular, we mined how long would a user generally browse for articles of a particular class of interest: starting from time $t = 0$ and averaging the times when an interest switch would take place. After this analysis we observed that on average, an article would be viewed for no more than 30 s for a successful 'hit' (i.e. an interesting article). Note however that this threshold really depends on the article's size and is simply a guideline based to our observations for the particular news articles dataset. In addition, from the same browsing behaviors, we observed that a user would spend no more than 10 min while viewing articles on our platform before he/she would leave the system altogether. This 'stream' of read articles during this period is a good indication of a user session.

```
Algorithm find_sessions
Input: history //time  window for sessions to be extracted
Output: Sessions[] //discovered sessions array
   viewing_threshold = 30 // at least 30 seconds
   session_threshold = 10 * 60 // at most 10 minutes
   previous_user = NULL
   current_session = NULL
   while fetch from DB (user, viewed article, timestamp, viewing_time) {
   if (viewing_time < viewing_threshold || timestamp < history)
       continue
   if (current_session.username != user) {
   // Since this is sorted by username, when a new user is found this means
   // a new session begins
       if (current_session.username!="" && current_session.articles !empty)
           Sessions[]+=current_session
       current_session.username = user;
       current_session.user_id = user_id;
       current_session.start = timestamp;
       current_session.articles.add(article_id);
   }
   else {
   // If the user is the same as before but the access time for this
   // article exceeds the time limit, a new session begins
       if (timestamp - current_session.start) > session_threshold ) {
           if (current_session.username!=""&&current_session.articles=!empty)
               Sessions[]+=current_session
           current_session.username = user;
           current_session.user_id = user_id;
           current_session.start = timestamp;
           current_session.end = timestamp;
           current_session.articles.add(article_id);
       }
       else {
           // The access time for this article does not exceed
           // the time limit
           current_session.articles.add(article_id);
          current_session.end = timestamp;
       }
 return Sessions[]
```

Algorithm 1. Discovering Sessions in user's access paths.

The output of Algorithm 1 is a list of sessions for each user that is stored in the database and used during the clustering stage later on.

4.2 Clustering user sessions

Once user sessions have been extracted, for each one of them, we aggregate the news articles that make up this session. At the next step, we enrich the keywords that belong to the session by using their hypernyms from the WordNet database and then proceed with session/user clustering via W-kmeans as explained in Bouras and Tsogkas [2]. The output of this step is clustered user sessions which can be thought of clusters of users.

As far as the WordNet hypernym extraction and keyword aggregation is concerned, for each given keyword of the session, we generate its graphs of hypernyms leading to the root hypernym (commonly being 'entity' for nouns). Following, we combine each individual hypernym graph to an aggregated one. For example, Fig. 3 depicts the aggregate hypernyms WordNet graph of the words 'pie', 'apple' and 'orange'.

There are practically two parameters that need to be taken into consideration for each hypernym of the aggregate tree-like structure in order to determine its importance: the depth and the frequency of appearance. It is observed that the higher (i.e. less deep, walking from the root node downwards) the hypernym is in the graph, the more generic it is. However, the lower the hypernym is in the graph, the less chances does it have to occur in many graph paths, i.e. its frequency of appearance is low. Note that in cases where a hypernym has multiple paths leading to a root, the shortest path is kept for representing its depth.

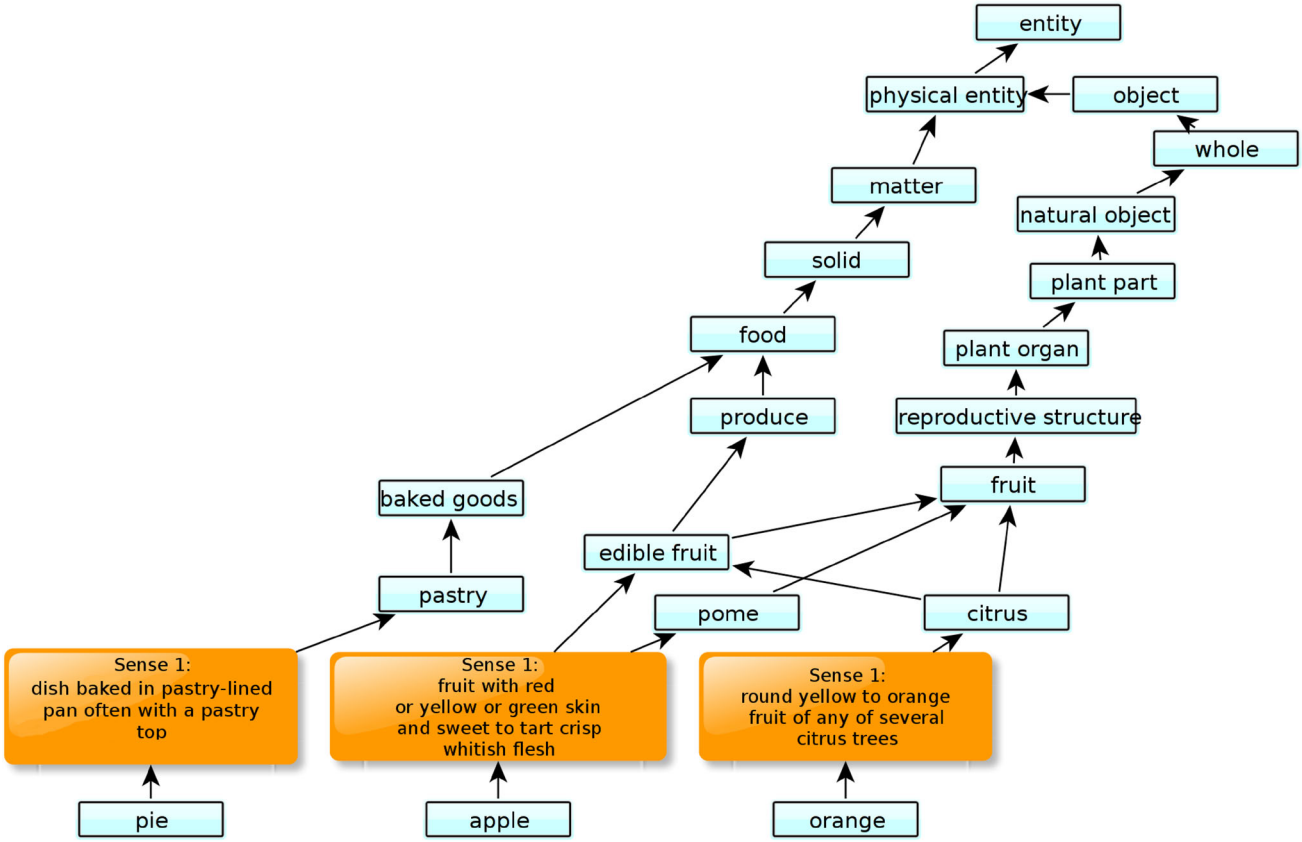


Fig. 3 Aggregate hypernym graph for three words: 'pie', 'apple', 'orange'

In our approach, those two contradicting parameters are weighted using (2).

$$W(d,f) = 2 \times \frac{1}{1 + e^{-0.125(d^3 \frac{f}{TW})}} - 0.5 \quad (2)$$

where d stands for the node's depth in the graph, f is the frequency of appearance of the node to the multiple graph paths and TW is the number of total words that were used for generating the graph (i.e. total keywords of the session). The Eq. (2) is a sigmoid one in the weighted form of: $a \times sig(d,f) - b$. We determined the best suited values for a and b via a simple experiment. Using a corpus of 1,000 pre-categorized news articles, we tried to determine the efficiency of the proposed W-kmeans algorithm via clustering these articles to the predetermined set of system categories. In this scenario, our clustering approach should make cluster assignments as close as possible to the categories of the articles. A variety of combinations were used and the best overall result was achieved with $a = 2$ and $b = 0.5$. The steepness value of (2) is affected by both the frequency and the depth of the hypernym. We chose a sigmoid function after observing how the depth and frequency affect the generated clustering results: the importance (weight) of each hypernym exhibits a progression from small beginnings that accelerates and approaches a climax over time, a behavior that is affected by the two previously mentioned factors. For large depth–frequency combinations, the weight of the hypernym reaches closer and closer to 1 (neither f nor d can be negative), whereas for low depth–frequency combinations the weight is close to 0. A keyword having no hypernym or not being in WordNet is omitted both from the graph and the TW sum. Furthermore, a hypernym may have multiple paths to the root, but is counted only once for each given keyword. Note also that the depth has a predominant role in the weighing process, much greater than frequency does. Frequency, however, acts as a selective factor when the graph expands with more and more keywords being added. We concluded to this weighing scheme after observations of hypernym graphs generated over hundreds of keywords because it scales well with real data. The aforementioned steps are summarized in Algorithms 2, 3. The process of user sessions clustering is continuously running in our system and as such user sessions do get clustered more than once in different clusters via different clustering passes. After that, for deciding which user sessions cluster to associate with a specific user, relatively 'fresh' clustering passes are only kept. Nevertheless, the above process

should not be misinterpreted as 'fuzzy' clustering, e.g. like in Wang et al. [35] and Ma et al. [21].

```
Algorithm clustering_user_sessions
Input: sessions, number of clusters
Output: session to cluster assignments
for each session s {
    for each article a belonging to s
        session.kws += fetch 20% most frequent k/ws for a
    wordnet_enrich(s) // See Algorithm 3
  }
clusters = kmeans(sessions)
return clusters
```

Algorithm 2. Clustering User Sessions using WordNet.

```
Algorithm wordnet_enrich
Input: session s
Output: session with enriched list of keywords
total_hypen_tree = NULL
kws = fetch 20% most frequent k/ws for s
for each keyword kw in kws {
    htree = wordnet_hypen_tree(kw) //extract the hypernym tree from WordNet
    for each hypen h in htree {
        if (h not in total_hypen_tree)
            h.frequency=1
            total_hypen_tree ->append(h)
        else
            total_hypen_tree ->at(h)->freq++
    }
}
for each h in total_hypen_tree {
    calculate_depth(h)
    weight = 2*((1/(1+ exp(-0.0125 *
                (h->depth ^3 * h->freq/ kws_in_wn->size)))) - 0.5))
}
sort_weights(total_hypen_tree)
important_hypens = (kws ->size/4)*top(total_hypen_tree)
return kws += important_hypens
```

Algorithm 3. Enriching user sessions using WordNet hypernyms.

4.3 User profiles and personalization using user clustering

Given a user u and a set of news articles R on which u provided, either implicitly or explicitly, a positive or negative feedback according to his/her interests (positive or negative, respectively), a user profile Up is maintained, analyzed by two parts. The positive part, U_p^+ consists of keywords from news articles judged positively by u, while the negative part, U_p^- consists of keywords from news articles judged negative judged by u. Moreover, each keyword is weighted via W_{kwi} depending to its ability to highlight the user's positive or negative preference. More formally,

$$U_p^+ = (kw_i \times W_p kw_i),\ \mathrm{i} = 1\ldots\mathrm{q}\ \text{with}\ \mathrm{q} <= |\ \mathrm{R}\ | \tag{3}$$

$$U_p^- = (kw_j \times W_n kw_j),\ \mathrm{j} = 1\ldots\mathrm{m}\ \text{with}\ \mathrm{m} <= |\ \mathrm{R}\ | \tag{4}$$

where kw_i the keyword in question, W_p the weight given to positive attributing keywords, W_n the weight given to negative attributing keywords.

The steps that are followed by the personalization procedure are presented in Algorithm 4. When a new user is registering to the system, he/she states the keywords of his/her preference as well as the scores that describe this preference initializing thus his/her profile. This procedure is trivial and can be avoided altogether since the personalization subsystem keeps track of the user's choices and browsing history, and so the user's preferences are updated on each visit. The user's profile consists of two keyword lists: a positive one, where the user-preferred keywords are placed, and a negative one where uninteresting keywords for the user are kept. By using these lists, we can personalize the news articles and summaries with satisfactory results.

```
Update_profile(a, b, c, d){
    Get_articles(a,b,d) //for factors a,b,d
        for each article{
            if (full article)
                if (time_viewed > Rar_thr1 && time_viewed < Rar_thr2){
                    Keywords_positive = top 5 frequent keywords
                    Update_list(Positive, Keywords_positive)
            }
            else {
            if(time_viewed> Rsum_thr1 && time_viewed< Rsum_thr2){
                Keywords_positive = top 5 frequent keywords
                Update_list(Positive, Keywords_positive)
            }
            Get_articles(c) //for factor c
            for each article{
                Keywords_negative = top 5 frequent keywords
                Update_list(Negative, Keywords_negative)
            }
Get_article(lists){
//Recovers the browsed articles and the amount
//of time spent reading the full article or its //summary (a,b).
//Recovers the articles with negative
//preference(c).
//Recovers the most frequently viewed articles
//by the user's cluster (d)
}
Update_list(list, keywords){
    for each (keyword in keywords)
        if (keyword not in list[])
            list.add(keywords[keyword])
        else
            list.update_freq(keywords[keyword])
}
```

Algorithm 4 Personalization steps for utilizing user feedback

The profile update procedure described in Algorithm 4, running constantly at every user's visit, takes note of the following aspects: (a) the browsed articles (the ones that the user selected to view), (b) the amount of time a user spends viewing the summary or the full text of a specific article, (c) the articles that the user avoids viewing (either their summary or their full text); the above derives from the simple logical assumptions that follow. In order to determine the above, some logical assumptions, also noticed into the user's browsing patterns follow:

1. A user will most likely spend an amount of time above a certain threshold, R_{ar_thr1} or R_{sum_thr1}, reading an article's full text or its summary, respectively, that is of interest for him/her (factor a).
2. However, an upper bound, R_{ar_thr2} and R_{sum_thr2}, should be used for these metrics since we do not want the mechanism to mistake forgotten browsed articles for the really interesting ones.

The thresholds that are used for R_{ar_thr1} and R_{ar_thr}, are selected to be 30 s and 3 min, respectively defining thus which article's keywords should be added (or have their weight increased) in the user's positive keywords list. We

concluded on the above threshold values after some inspection on the recorded user preferences in the databases since in most cases, when a user would spend an amount of time less than those thresholds (for either the full article or its summary) he would act upon this article (e.g. follow its source link).

The summary viewing thresholds are calculated in an analogous way:

$$R_{sum\_thr1} = R_{ar\_thr1} * S_{ratio} \quad (5)$$

$$R_{sum\_thr2} = R_{ar\_thr2} * S_{ratio} \quad (6)$$

where S_{ratio} expresses the summarization "compression ratio":

$$S_{ratio} = \frac{\#words(summary)}{\#words(fulltext)} \quad (7)$$

Moreover, most of the times, a user will select to browse articles of a topic that he/she finds interesting (factor b) as advertised by the article's title and/or summary. Lastly, a user will probably avoid visiting articles that he/she finds uninteresting and thus the keywords that represent those articles should be receiving a lessened or negated weight (factor c).

In addition to the above factors (a–c), having deduced the user's cluster by following the steps described in Sect. 4.2 via W-kmeans, we can also take the user clustering information into account. More specifically, from the cluster the user belongs to, we can enrich the user's positive keywords list using articles that have been frequently viewed by the cluster members (at least by 20 % of the cluster users). From those articles we keep the top five keywords which have also been previously enriched by WordNet. We call this user clustering heuristic: factor d.

From the above factors, the personalization algorithm keeps track of the keywords that the user has expressed preference to, combined with similar preferences of people from the same cluster, and thus, the articles (containing these keywords) that he/she will likely be willing to read in the future. The parameter that depicts the user's preference for a keyword according to the aforementioned factors (a–d) is U_{wi} and is based on the relative frequency that the keyword has on the list, a frequency that is constantly modified by the user's choices. U_{wi} is derived from the following equation:

$$U_{wi} = rel(fr(_{kwi})) \times (1 + T_{kwi}) \quad (8)$$

where $rel(fr(_{kwi})$ is the relative frequency of keyword i, T_{kwi} is the normalized total time spent on the specific keyword if it belongs to the positive list; however if the keyword is in the negative list, T_{kwi} is set to 0 since no time is actually spent on these keywords by the user. In case the keyword is originating from the user clustering process and thus has not been explicitly preferred by the user, we average on the total amount of time the users of the cluster spend on the article this keyword originates from. Furthermore, we expect that when the user profile reaches its steady state, the mean times of the keywords preferences to be correct, hence depicting the overall user preferences. The overall personalization factor for each keyword i, named U_{pi}, is:

$$U_{pi} = B \times U_{wi} \quad (9)$$

where, for the parameter B: if the keyword belongs to the positive keyword list, then $B > 1$; whereas if the keyword belongs to the negative keyword list, then $B < 1$. The norm of the B parameter can take any value that we desire, thus increasing or decreasing at will the effect that personalization and dynamic profile generation have on the sentence weighting procedure. From the previous, Up_i can be positive, negative or zero if there is no information about the user's preference of the specific keyword.

After the overall personalization factor for each keyword i is generated, the list of keywords is sorted in descending order. Following that, the recommendations for each user are created based on the articles that contain the most and highly weighted keywords of the list.

5 Experimental procedure

In this section we describe the experimental procedure that was followed in order to evaluate our user clustering approach and the effect that it has on personalizing user preferences. We present the used corpus and the evaluation metrics used as well as the actual experimental results.

5.1 Corpus and user browsing behavior

For our experimentation we analyzed the logs of the browsing patterns as well as the recommendations offered to 50 of the registered system users. The users had been using the system for 2 months after their registration and for this period of time, the recommendations with and without the application of user clustering via W-kmeans were tracked. The total amount of articles recommended or browsed, i.e. the used corpus was over 8,000 articles which belonged to various fields of interest: politics, technology, sports, entertainment, economy, science and education. Those articles were fetched randomly from the headlines (i.e. 'top stories') of major news portals from the web, including bbc.com, cnn.com, reuters.com and espn.com. As such, these articles are representative of, as well as balanced on, each of the above domains of user interests.

Once the articles were fetched, and after the preprocessing procedure and most notably stemming and noun

identification, we kept for each article its list of stemmed nouns. Notice that duplicate articles originating from different sources were detected based on their title and main body and removed from the dataset. We also used the navigational patterns that we recorded for the 50 registered system users at the same period which were depicted by either of the following cases: (a) articles suggested by the system and selected for viewing by the user (based on the thresholds described in Sect. 4), (b) articles not suggested by the system but viewed by the user, and (c) articles suggested by the system but deliberately not viewed by the user (or viewed for a very short amount of time).

5.2 Evaluation metrics

For our evaluation metrics we used the clustering index (CI), the mean absolute error (MAE) and the F-measure. In order to determine the efficiency of each clustering pass, together with the right number of clusters for your dataset, we used the evaluative criterion of CI, as defined in formula (10).

$$CI = \bar{\sigma}^2/(\bar{\sigma} + \bar{\delta}) \tag{10}$$

where $\bar{\sigma}$ is the average intra-cluster similarity and $\bar{\delta}$ is the average inter-cluster similarity. Intuitively, since the most efficient clusters are the ones containing articles close to each other (within the cluster), while sharing a low similarity with articles belonging to different clusters, good CI results should focus on increasing the first measure (intra-cluster similarity) while decreasing the second (inter-cluster similarity).

Another widely used evaluation metric for predicting performance of CF and recommender systems is MAE. MAE, expresses the average absolute deviation between predicted and true ratings and can be computed using formula (11).

$$MAE = \frac{\sum r'(u,i) \in R' |r(u,i) - r'(u,i)|}{|R'|} \tag{11}$$

where: r(u, i) is the preference of user u for article i and r'(u, i) the predicted/recommended preference for user u of articles belonging to R'. For an CF/recommender approach to provide good results the MAE scores should be as low as possible.

Our third evaluation metric, the F-measure, as defined in (12) is a weighted combination of the precision and recall metrics and is employed to evaluate the accuracy and efficiency of our recommendation system when using user profile clustering. We define a set of target articles, denoted C, that the system suggests and another set of articles, denote C', that are visited by the user after the recommendation process. Moreover, doc (c', c) is used to denote the number of documents both in the suggested and in the visited lists.

$$F(c',c) = 2 \cdot \frac{r(c',c)p(c',c)}{r(c',c) + p(c',c)} \tag{12}$$

where $r(c',c) = \frac{doc(c',c)}{doc(c')}$
and $p(c',c) = \frac{doc(c',c)}{doc(c')}$

5.3 Experimental results

For our first experiment we compared W-kmeans to standard k-means when applied to user clustering. The results, depicted in Fig. 4, show that W-kmeans clearly outperforms standard k-means by at least a factor of 10, thus providing clusters of users more tightly bound. Consequently, the generated clusters can capture with a better accuracy users with similar interests while successfully separating users with contradicting interests. From Fig. 4 it can also be deducted that both of the CI graphs peak at around 30 clusters. This is a good indication about the best suited amount of clusters applicable for our dataset, a finding that shows that the W-kmeans algorithm generates fine-grained clustering results.

Figure 5 depicts the MAE results that we obtained during this experimental procedure. As shown from it, the application of article and user clustering via the W-kmeans algorithm has significantly reduced the MAE of the recommendations provided to the users. More specifically, we observed that as users were viewing more and more articles and their profiles got shaped, the MAE of the recommendations reduced. This was true both when user clustering was applied and when not applied. What the above means from the practical point of view is that the recommendations given to the users were, with increasing tendency, accurate since users opted on viewing them. A similar result, which had to do with news articles clustering though, has also been previously observed in Bouras and Tsogkas [2]. However, by taking into account the user clustering information, the MAE of article suggestions

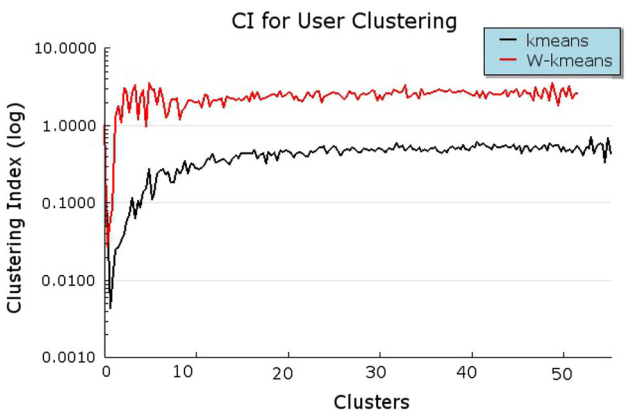


Fig. 4 Comparison of W-kmeans and k-means for user clustering

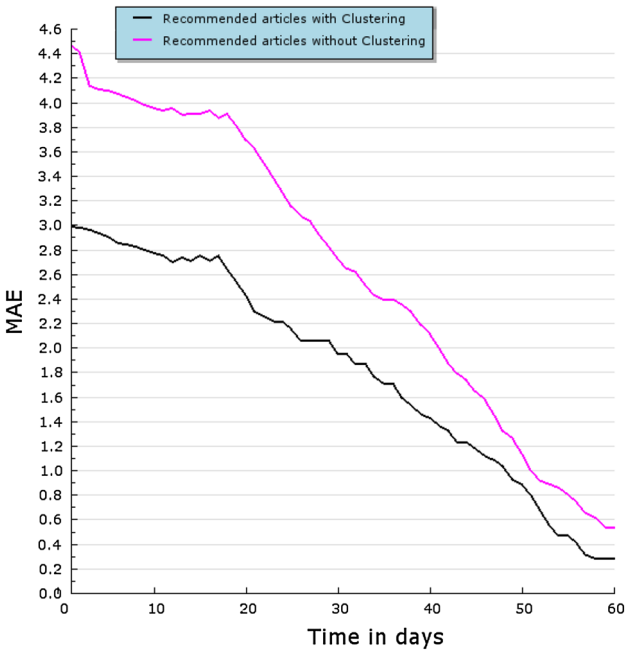


Fig. 5 MAE of recommendations with and without the use of W-kmeans

compared to the actual user choices reduced by an average of 15 % over the case when user clustering was not applied. This was more obvious at the early days of system usage during the experiment, when the user profiles had not yet been determined to a good extent by our system. Nevertheless, even when the user profiles had on average reached a 'steady state', around day 45 (as observed from the average numbers of profile updates), the MAE was still less when clustering was taken into consideration by the recommendation process, proving in effect the significance of our approach.

For our second experiment we tried to determine the overall improvement of our recommendation engine when taking into consideration existing user clusters. As explained in Algorithm 4, for returning users we modified our recommendation stage to suggest 10 of the top viewed articles belonging to the user's cluster. Following, we recorded which of the suggested articles were viewed by the user within a time frame of 30 min. The process was repeated without the user clustering enhancement of the recommendation engine but with other heuristics, such as text categorization and personalization still enabled. The results, presented in Fig. 6 show the average F-measure for each case as users increase.

From Fig. 6, we observe an average improvement of 10 % with regards to the F-measure when user clustering is deployed. The efficiency also rapidly increases as more users are taken into consideration by the system, something that is expected, given the personalization features of our

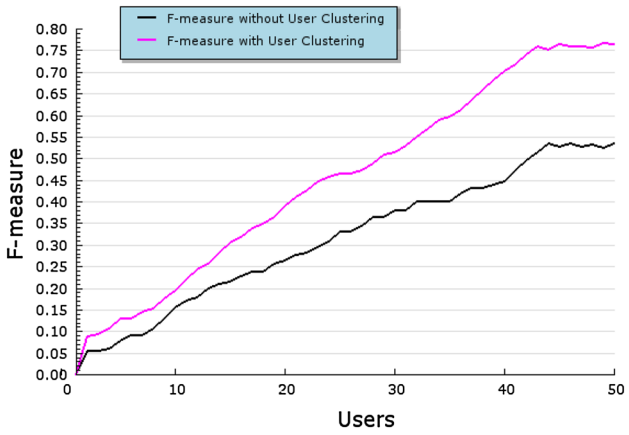


Fig. 6 Comparison of the recommendation engine performance

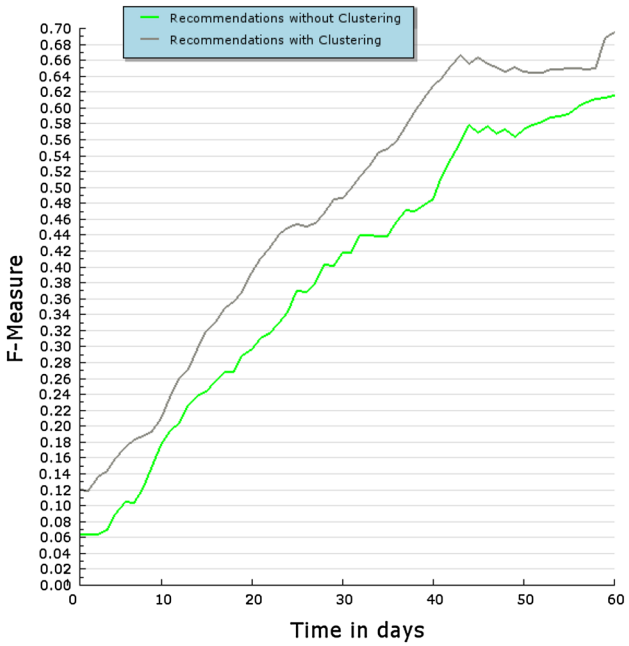


Fig. 7 F-Measure of recommendations with and without the use of W-kmeans

recommendation engine. From a natural point of view, our experiments showed that the resulting suggestions matched the user's choices in average 7 out of 10 times. In our opinion this proves that our approach has greatly benefited the recommendation stage. Another observation from Fig. 6 compared to Fig. 5, is that we can see some specifics regarding how many users does it take for our recommendation engine to suggest 'useful' recommendations to the returning users—thus improving the overall F-measure results.

Using the same user logs as in the previous experiment and for the same time window, we extracted the F-measure results for the produced recommendations as the days passed depicted in Fig. 7.

From the graph of Fig. 7, we observe that the recommendations which utilize the generated article and user clusters produce on average 0.1 better scores in terms of F-measure. As before, the improvement gets even better after some days of system usage. The above has two explanations: (a) the system has more data regarding the users' choices/preferences, and (b) the system has more time to generate more coherent and generally better user clusters. At first the F-measure scores are too low because of the fact that the recommender hasn't yet determined the user profiles to an acceptable extend. What is also observed from the previous graph is that around day 45 the recommendations have reached almost at their performance peak revealing that on average, the steady state for the users' profiles has been achieved.

For our last experimentation procedure, we tried to compare the efficiency of the proposed methodology versus some state of the art CF methods, like latent semantic CF, neighbor-based CF and dimensionality reduction techniques like SVD. For the above methodologies we have used the open source library from [17]. This experiment was also executed on the previously described dataset and for the same users. The results, presented in Table 2, revealed that W-kmeans outperformed or was at least as equal as those methods in terms of F-measure average over various users. Notice however that even though other CF methodologies might have yielded similar performance in terms of article suggestions to the users, the execution times of the proposed methodology was significantly less as shown in Table 3. The only exception was SVD were even though the execution times were significantly less, there are also two factors that should be factored in for our comparison: the initial dimensionality reduction computing times over our data (almost 20 s that is paid once), and the inability to easily incorporate (for pure SVD) added data. Especially the latter, even though of major significance for an online CF system, is not captured by our experimentation.

Table 2 CF methodologies comparison

CF methodology	Average F-measure over all users
W-kmeans	0.45
Latent semantic CF	0.4
Neighbor-based CF	0.3
Dimensionality reduction (SVD)	0.45

Table 3 Execution time comparison

CF methodology	Execution time (s)
W-kmeans	12
Latent semantic CF	20
Neighbor-based CF	25
Dimensionality reduction (SVD)	2

6 Conclusions

In this paper we used the WordNet-enabled k-means algorithm, which explores the usage of word hypernyms extracted from the WordNet database, to the field of profile clustering as well as its application to our recommendation system. Trying to deal with the task of effective and adequate retrieval of personalized news articles that derive from the web, we presented the personalization algorithm that is used for presenting the categorized, clustered and summarized articles to the user. Our recommendation approach can be classified as 'hybrid' since it is mainly content-based with some collaborative filtering features that enhance the algorithm with the ability to automatically adopt over time to the continuously changing user choices.

We examined the performance of this approach compared to standard k-means and discovered a 10-fold amelioration in terms of cluster coherence. Furthermore, we found an average improvement of around 10 % in terms of F-measure for the resulting suggestions of our recommendation engine when used by real system users. Additionally, some basic experimentation showed that W-kmeans performs usually better compared to other CF techniques when applied to our recommendation system. Our experimentation also showed a significant MAE diminution, on average 15 %, when clustering was applied before the recommendations instead of when not using it. We also noticed that the recommendations were scoring on average 0.1 better in terms of F-measure.

We believe that the above results justify the use of clustering, both article and user based, in a recommendation system since they seem to benefit it significantly.

7 Future work

For the future, we are planning on enriching the various components of our system with improved techniques. More specifically, for keyword extraction we plan on enriching the extracted keywords with the use of online thesauri. For article categorization we plan on applying various more methodologies that may yield better results depending on the dataset.

Additionally, given the fact that average error measures like MAE or F-measure miss the features that are the most important for user satisfaction and new notions of quality evaluation have emerged, we are planning on adjusting our approach based on other criteria, such as: avoiding bad mistakes, diversifying the responses, extending on

multidimensional ratings (or tags) and fine tuning direct and indirect user ratings. Besides, error and correlation scores might do a good job at testing recommendations as an approach to recover missing data, but do not work well at assessing if they convey valuable items previously not known to the user—something for which the recommenders are designed in the first place.

We are also planning on incorporating cluster labeling for the generated profile clusters to the system, as well as automate the detection of the best suited number of clusters for W-kmeans that is best for the underlying data. Furthermore, we are focusing on creating suitable communication channels for delivering the article recommendations to the user's desktop or handheld device.

Acknowledgments

This research has been co-financed by the European Union (European Social Fund—ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF)—Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

References

1. Ansari A, Essegaier S, Kohli R (2000) Internet recommendation systems. J Mark Res 37(3):363–375
2. Bouras C, Tsogkas V (2010) W-kmeans: clustering news articles Using WordNet. In Proceedings of KES, 3, pp 379–388
3. Bouras C, Poulopoulos V, Tsogkas V (2008) PeRSSonal's core functionality evaluation: enhancing text labeling through personalized summaries. Data Knowl Eng 64(1):330–345
4. Cadez I, Heckerman D, Meek C, Smyth P, White S (2000, August) Visualization of navigation patterns on a web site using model-based clustering. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 280–284
5. Chee SHS, Han J, Wang K (2001) Rectree: An efficient collaborative filtering method. In: Data Warehousing and Knowledge Discovery, pp 141–151
6. Cooley R, Mobasher B, Srivastava J (1999) Data preparation for mining world wide web browsing patterns. Knowl Inf Syst 1(1):5–32
7. Eirinaki M, Vazirgiannis M (2003) Web mining for web personalization. ACM Trans Internet Technol (TOIT) 3(1):1–27
8. Ekstrand MD, Riedl JT, Konstan JA (2011) Collaborative filtering recommender systems. Found Trends in Hum-Computer Interact 4(2):81–173
9. Fu Y, Sandhu K, Shih MY (1999) Clustering of web users based on access patterns. In Proceedings of the 1999 KDD Workshop on Web Mining. San Diego, CA
10. Hannon J, Bennett M, Smyth B (2010, September) Recommending twitter users to follow using content and collaborative filtering approaches. In Proceedings of the fourth ACM conference on Recommender systems. ACM, pp 199–206
11. Hay B, Geert W, Koen V (2011) Clustering navigation patterns on a website using a sequence alignment method. Intelligent techniques for web personalization: IJCAI (2001): 1–6
12. Hofmann T (2004) Latent semantic models for collaborative filtering. ACM Trans Inf Sys (TOIS) 22(1):89–115
13. Joachims, T. (2002, July). Optimizing search engines using clickthrough data. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 133–142
14. Konstan JA, Riedl J (2012) Recommender systems: from algorithms to user experience. User Model User-Adap Inter 22(1–2):101–123
15. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
16. Koren Y, Bell R (2011) Advances in collaborative filtering. In: Recommender systems handbook, pp 145–186
17. Latent semantic analysis—open source open source implementation. https://code.google.com/p/airhead-research/wiki/LatentSemanticAnalysis
18. Li C, Li J, He M (2014) Concept lattice compression in incomplete contexts based on K-medoids clustering. Int J Mach Learn Cybern, 1–14
19. Li Y, Chung SM (2007) Parallel bisecting k-means with prediction clustering algorithm. J Supercomput 39(1):19–37
20. Lops, P., Degemmis, M., & Semeraro, G. (2007). Improving social filtering techniques through WordNet-Based user profiles. In: User modeling 2007, pp 268–277
21. Ma W, Jiao L, Gong M, Li C (2014) Image change detection based on an improved rough fuzzy c-means clustering algorithm. Int J Mach Learn Cybernet 5(3):369–377
22. Maier M, Hein M, von Luxburg U (2009) Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. Theoret Comput Sci 410(19):1749–1764
23. Melville P, Mooney RJ, Nagarajan R (2002) Content-boosted collaborative filtering for improved recommendations. In: AAAI/IAAI, pp 187–192
24. Miyahara K, Pazzani MJ (2000) Collaborative filtering with the simple Bayesian classifier. In: PRICAI 2000 Topics in Artificial Intelligence, pp 679–689
25. Mobasher B, Cooley R, Srivastava J (2000) Automatic personalization based on Web usage mining. Commun ACM 43(8):142–151
26. Pan R, Dolog P, Xu G (2013) KNN-based clustering for improving social recommender systems. In: Agents and data mining interaction, pp 115–125
27. Qin M, Buffett S, Fleming M (2008) Predicting user preferences via similarity-based clustering
28. Pavlov DY, Pennock DM (2002) A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In: Advances in neural information processing systems, pp 1441–1448
29. Revankar P, Dahiwele J (2011, March) Web Usage Mining. In: 5th National conference, pp 10–11
30. Shani G, Brafman RI, Heckerman D (2002) An MDP-based recommender system. In Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence, pp 453–460
31. Si L, Jin R (2003) Flexible mixture model for collaborative filtering. ICML 3:704–711
32. Tang N, Vemuri VR (2005) User-interest-based document filtering via semi-supervised clustering. In: Foundations of intelligent systems, pp 573–582
33. Ungar LH, Foster DP (1998, July) Clustering methods for collaborative filtering. In: AAAI workshop on recommendation systems, Vol 1
34. Varelas G, Voutsakis E, Raftopoulou P, Petrakis EG, Milios EE (2005, November) Semantic similarity methods in wordNet and

their application to information retrieval on the web. In Proceedings of the 7th annual ACM international workshop on Web information and data management. ACM, pp 10–16

35. Wang X, Wang Y, Wang L (2004) Improving fuzzy c-means clustering based on feature-weight learning. Pattern Recogn Lett 25(10):1123–1132

36. Yeung DS, Wang XZ (2002) Improving performance of similarity-based clustering by feature weight learning. Pattern Anal Mach Intell IEEE Trans 24(4):556–561