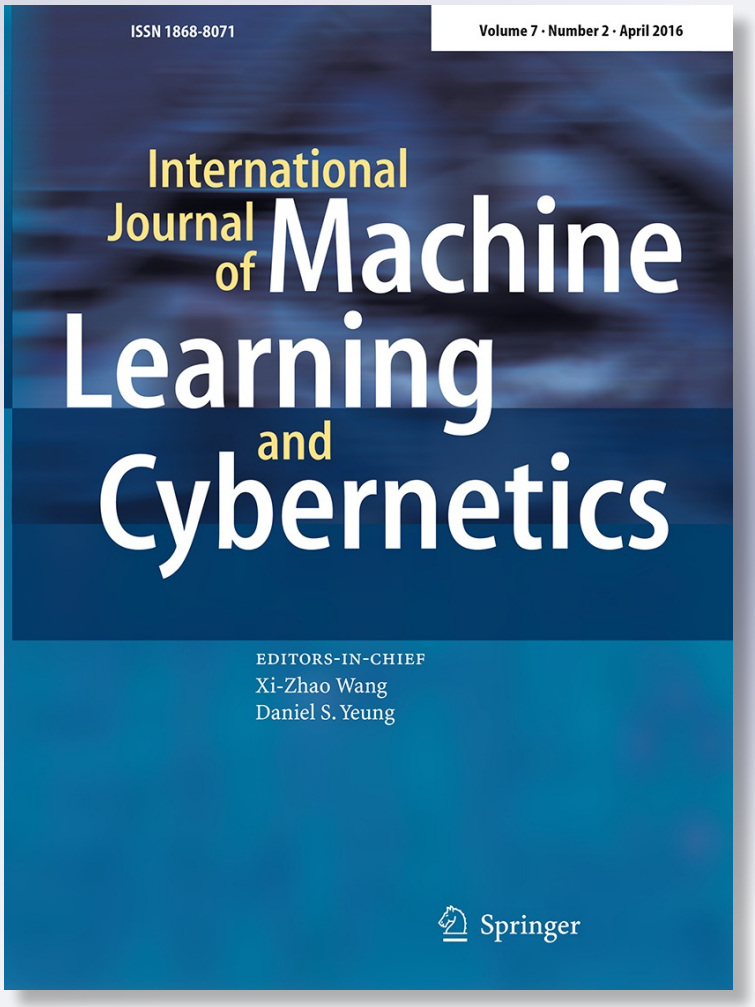
Assisting cluster coherency via n-grams and clustering as a tool to deal with the new user problem

Christos Bouras & Vassilis Tsogkas

International Journal of Machine Learning and Cybernetics

ISSN 1868-8071
Volume 7
Number 2

Int. J. Mach. Learn. & Cyber. (2016)
7:171-184
DOI 10.1007/s13042-014-0264-y



Springer

Int. J. Mach. Learn. & Cyber. (2016) 7:171–184
DOI 10.1007/s13042-014-0264-y

ORIGINAL ARTICLE

Assisting cluster coherency via n-grams and clustering as a tool to deal with the new user problem

Christos Bouras · Vassilis Tsogkas

Received: 12 December 2013 / Accepted: 29 April 2014 / Published online: 11 May 2014

Abstract Collaborative filtering systems typically need to acquire some data about the new user in order to start making personalized suggestions, a situation commonly referred to as the "new user problem". In this work we attempt to address the new user problem via a unique personalized strategy for prompting the user with articles to rate. Our approach makes use of hypernyms extracted from the WordNet database and proves to be converging fast to the actual user interests based on minimal user ratings, which are provided during the registration process. In addition, we explore the possible enhancement of the document clustering results, and in particular clustering of news articles from the web, when using word-based n-grams during the keyword extraction phase. We present and evaluate a weighting approach that combines clustering of news articles derived from the web, using n-grams that are extracted from the articles at an offline stage. This technique is then compared with the single minded "bag-of-words" representation that our clustering algorithm, W-kmeans, previously used. Our experimentation reveals that via fine tuning the weighting parameters between keyword and n-grams, as well as the n value itself, a significant improvement regarding the clustering results metrics can be achieved.

C. Bouras · V. Tsogkas
Computer Engineering and Informatics Department, University of Patras, Patras, Greece
e-mail: tsogkas@ceid.upatras.gr

C. Bouras (✉)
Computer Technology Institute and Press "Diophantus", Rion, 26500 Patras, Greece
e-mail: bouras@cti.gr

Keywords New user problem · Collaborative filtering · Clustering · W-kmeans · K-means · Personalized strategy · n-grams · Text preprocessing

1 Introduction

Every day, more and more news articles, books, journals, research papers, web pages, and movies are being made available online. While available information is growing in volumes, we quickly become overwhelmed and seek assistance in finding the most interesting, valuable, or entertaining items on which we should spend our scarce time. Historically, humans have adapted well to pieces of information and have developed an excellent filtering ability to make quick judgments.

The technologies that are commonly used to address the previously mentioned information overload challenges are basically three. Each one of them focuses primarily on a particular set of tasks or questions:

- Information Retrieval (IR), which focuses on tasks involving fulfilling ephemeral interest queries, such as finding the articles related to president Obama
- Information Filtering (IF), which focuses on tasks involving classifying streams of new content into categories, such as finding any newly released articles regarding the political situation in Middle East, or any newly released movies without an English-language soundtrack or subtitles (to reject)
- Collaborative Filtering (CF), which focuses on two important questions: which items (from a set or overall) should be proposed to a user, and how appealing these particular items will be for the user.

Each of the above technologies has a specific approach in producing an effective recommender system.

Clustering, i.e. the task of grouping a set of objects in such a way that objects of the same group are more similar to each other than to those of other groups, plays a central role in data analysis. It can give useful information about the structure of the underlying data. By discovering interesting information kernels and distributions, it has proven to be a key technique for information retrieval (IR) and the decision making process. Clustering also plays a crucial role in organizing large collections and can be used (a) to structure query results, (b) form the basis for further processing of the organized topical groups using other information retrieval techniques, such as summarization, or (c) within the scope of recommendation systems, by affecting their performance as far as suggestions made towards the end user are concerned.

Achieving good clustering performance by improving cluster coherency is a major goal for all the related literature in recent years. Being a domain-dependent technique, a number of heuristics have been devised that apply to particular fields of interest but cannot be generally adopted. For example, techniques for protein clustering have little application to the domain of news articles clustering.

Another commonly faced problem (also known as 'the new user problem') that collaborative filtering techniques strive to deal with is the situation in which a new and generally unknown individual is requesting information from a system for the first time. In this scenario, little to no personalization can be done for the particular user, something that severely affects the system's recommendation performance towards that user.

Our research is targeted to the domain of news articles originating from the web. In this manuscript we are trying to deal with the two afore-mentioned problems that modern indexing systems suffer from: (a) poor clustering performance and, (b) poor personalization results due to the "new user problem".

The rest of the manuscript is structured as follows: Sect. 2 gives an overview of the related work regarding the latest developments in the field of document clustering, the use of n-grams for improving cluster coherence, as well as methodologies developed for alleviating the "new user problem". In Sect. 3 we are presenting the information flow suggested by our approach. Section 4 gives the algorithmic steps developed for dealing with the two problems analyzed in our work. In Sect. 5 we present the experimental results, while in Sect. 6 we conclude this manuscript. Section 7 gives some pointers for future research on the particular field.

2 Related work

2.1 Clustering as a collaborative filtering technique

Collaborative filtering is a methodology with which users co-operate, usually in an implicit manner, in order to determine what is interesting or useful from a large set of items. The intuition behind this approach is that since people prefer only to look at relevant, interesting content, one way to filter out the irrelevant data is to collaborate with other users and consider what they believe to be relevant [19].

A good analysis of how recommender systems spring up and evolved over time is given in the extensive work of Ekstrand et al. [11]. Nowadays, there are many more CF systems online (Amazon, Google News, YouTube, Netflix, etc.) most of which generate recommendations to the user, based on those items that have the highest predicted rating among those that the user has not yet rated. The previous technique makes CF and recommender systems very similar.

Two generic categories of the various clustering methods exist in the literature: hierarchical and partitional. Typical hierarchical clustering techniques generate a series of partitions over the data, which may run from a single cluster containing all objects to n clusters each containing a single object, and are widely visualized through a tree-like structure. On the other hand, partitional algorithms typically determine all clusters at once. For partitional techniques, a global criterion is most commonly used, the optimization of which drives the entire process thus producing a single-level division of the data. A typical partitional algorithm is k-means which is based on the notion of the cluster center, a point in the data space, usually not existent in the data itself, which represents a cluster. The family of k-means partitional clustering algorithms [31] usually tries to minimize the average squared distance between points in the same cluster. Several improvements have been proposed to the k-means algorithm in order to address its shortcomings. For example, Rana et al. [24] presented the BR-APSO approach for dealing with the starting conditions/local minima problem of k-means. Recently, Sarma et al. [27] also presented a hybrid approach that speeds-up the k-means clustering method significantly.

2.2 Not knowing the user

A common problem that all CF filtering systems suffer from is the cold start problem. It consists of a family of three related problems: (a) the new item problem, where a new item is introduced to the system and since it has no rating, the system cannot recommend it to any user, (b) the

new user problem, where a new user enters the system and there are no ratings made yet by him/her—hence there can be no generated system predictions for him/her, and (c) the new system problem, where we have a new system for which there are no ratings of any users. In this manuscript, we will be focusing on the new user problem.

Previous approaches to the new user problem have mostly focused on metadata and user-prompting. The metadata about items can be used to generate recommendations by content-based recommender systems like the one described by Balabanovie and Shoham [3], or in a hybrid fashion with ratings based system as the ratings come into the system [16]. Filterbots [22] constitute another approach in which pseudo-users and items are algorithmically created in an attempt to provide baseline ratings to the system, such that no user or item would be left without a rating. These agents, as shown by Good et al. [15], can potentially be performing better when working in tandem with CF techniques and in particular, the CF engine is what matters the most in this combinatory scenario. Other methods that utilize demographic data available to the system have also been proposed, but gathering such data often conflicts with privacy issues.

Recommender systems have also been used to help tackle the new user problem. Some approaches, as described by Nguyen and Haddawy [21], create user categories where new users are quickly assigned by using a set of predetermined questions. These approaches jump-start the system by using demographic or model based attributes. Even though somewhat domain-limited, they can potentially produce accurate results.

Another method for dealing with the new user problem is to explicitly ask the users to provide ratings for items (i.e. news articles in our case). The scheme is pretty basic: when the new user enters the system, he/she is presented with items to rate, which are not really recommendations, but are rather selected in order to gather as much information about his/her profile as possible. As he/she continues providing ratings for the selected items, the system decides whether to continue this process improving the user's profile or halt it. Large questionnaires come with a cost though: users are easily disturbed and might give up the process if it's taking too long or if they feel that the requested data conflict with their privacy. When this process ends, the system, having a basic knowledge of the user's appetite, starts recommending items and monitors his/her actions forming a feedback loop for profile updates.

This prompting approach was introduced by Kohrs and Merialdo [18], where the ordering of items by the variance and entropy were investigated. Methodologies regarding user prompting can be divided into non-personalized and personalized [9]. Non-personalized methodologies for user prompting include: (a) the popularity method, where items are ordered by the number of ratings that they have been given by all users, (b) the entropy and its variations methods that rely on the fact that certain items can yield more information about a user's likes than others, (c) the greedy method, where the next item is chosen from those that the user is able to rate, such that the prediction error for his/her test set is minimized (clearly this method could not be used in practice as it requires knowing not only what each person is able to rate but also the actual ratings), (d) other people's greedy and variations, where the items that will be presented to the user are chosen from the top-n lists of other users. Personalized methodologies, on the other hand, take into consideration the responses that the user has given to items already presented. Some non-personalized methodologies are: (a) naïve Bayes, where by knowing whether a user is able to rate an item we can work out the naïve Bayes probability of a user being able to rate the other items, (b) perturbed other people's greedy and variations which combine other people's greedy with naïve Bayes.

Rashid et al. [26] presented more methods for improving the order of items attempting also personalized orderings. Recently, a new method for a non-personalized ordering of items was presented by Golbandi et al. [13] and a personalized one by Golbandi et al. [14]. Pilaszy and Tikk [23] used a prediction method which was a variant of matrix factorization and they showed that more accurate predictions can be made when the user has provided minimal ratings than when the system uses the metadata of the items in order to generate predictions.

There are two utterly important aspects for the user prompting approach: (a) which items to select and (b) in which order to present them to the user. There have been many approaches regarding the process of selecting which items to present next to the user during the prompting phase. Certain trade-offs should also be considered, like the effort the user has to put into and the satisfaction he/she will get by the registration process as a whole. Moreover, the recommendation accuracy, i.e. how good the recommendations presented to the user really are, is of great significance. Taking the above into consideration, there have roughly been five main strategies for selecting which items to present to the user during the prompting phase: random, popularity, pure entropy, balanced and personalized [25].

In random strategies, the items that are to be presented are chosen randomly with a uniform probability over the universe of items. If the distribution of ratings is uniform, they have the advantage of covering the entire universe of items. In the popularity based strategies, the items are sorted in descending order based on their number of ratings. Even though they are easy to compute, these strategies overly promote items that have been widely rated and

may carry little information. In pure entropy strategies the users are asked about items which give the most information for each rating. Generally, an item that has some people who disliked it and some other who liked it may tell us more than an item which everybody liked. In balanced strategies, a combination of the popularity and entropy strategies is used. This can usually be in the form of Popularity * entropy or Log Popularity * Entropy. This approach, using Bayes theorem, silently assumes that popularity and entropy are independent which is not always correct. Lastly, in personalized strategies the suggestions are adapted to the user preferences via a feedback loop. Frequently, the item by item strategy is used in which, at first, items are presented to the user by any other strategy until a rating is picked up. Following that, a recommender, based on some similarity measure, finds similar items for user suggestions based on what the user previously rated. Yeung and Wang [29], showed that, by using a gradient descent technique to learn the feature weights, the clustering performance can be significantly improved, enhancing thus the quality of the similarity-based decision making. A similar result was shown by Wang et al. [28] for the case of feature weights regarding the performance of fuzzy c-means clustering.

In our work we are focusing on a personalized methodology for user prompting, similar to the item by item strategy. Our approach exploits clustering, and in particular our W-kmeans clustering algorithm [6], in order to select which news articles to pick next for rating by the user. Our work explores both item (i.e. article) and user clustering in order to effectively select articles for rating and thus can quickly and reliably converge to the actual user's preferences. We also try to determine how well our approach deals with the new user problem compared with the basic five strategies that were previously described.

2.3 N-grams for assisting news articles clustering

An n-gram is defined either as a textual sequence of length n, or similarly, as a sequence of n adjacent 'textual units', in both cases extracted from a particular document. A 'textual unit' can be identified at a byte, character or word level depending on the context of interest. In this work, we are dealing with word n-grams which can be conceptualized by placing a small sliding window over a sentence of a given text, in which only n words are visible at a given time. At each position of the window, the sequence of words inside it is recorded. In some schemes, the window may be slid more than one word after each n-gram is recorded. The simplest n-gram is the so-called unigram, where $n = 1$, which falls back to the single minded "bag-of-words" (BOW) representation. Typically, n is a fixed number, highly dependent on the particular corpus of documents and the queries made against that corpus. Each of the n-grams is a coordinate in a vector which represents the text under study and the frequency that this n-gram appears in the text can be the number of this coordinate. We can hence use this representation for text comparison, and as such, it can find many uses in various information retrieval (IR) tasks, including but not limited to item clustering.

The use of n-gram probability distribution and n-gram models in NLP is a relatively simple idea, but it has been found to be effective in many applications. For example, character level n-gram language models can be easily applied to any language, and even non language sequences such as DNA and music. They are also widely used in text compression, e.g. the PPM model [8], and have also been found to be effective in text mining problems as well [20]. In the domain of language independent text categorization word-based language modeling techniques were used for both English and German with good results [10]. N-grams analysis has proven of great significance in many areas of natural language processing and text mining, such as text parsing or information retrieval applications. Typical examples include a) searching and categorization of similar documents like in Cavnar and Trenkle [7], where the authors present a character n-gram-based approach to text categorization. They conclude that by using character n-grams, they are able to reliably categorize documents in a wide range of classification tasks, (b) identification of reused, duplicated or plagiarized text [2], (c) malicious code detection [1], and (d) a variety of linguistic tasks such as speech recognition (Jurafsky et al. [17]). The intuition behind all of the afore-mentioned approaches is common: phrases as a whole should carry more information than the sum of their individual components, extraction of which should lead to better textual representations and results.

Another aspect of n-gram analysis that should also be stressed out, is that infrequent n-grams are uninteresting, thus one only needs to keep track of n-grams that occur with frequency above a certain threshold—say at least m times. Furthermore, determining the correct value of n, i.e. the size of the sliding window that is to be used, when using word based n-gram analysis, is an area of experimentation on each particular domain of knowledge. For example, on the domain of plagiarism detection, Barron-Cedeno et al. [2] explain that low values for n appear to give the best results with a specific precision-recall trade-off. Values larger than four diminish the performance of the approach. A similar result is also pointed out in Furnkranz [12], who concludes that word sequences of length two or three prove to be the most useful since larger sequences reduce classification performance.

In our previous work [5], we proposed a new clustering method, called W-kmeans, which improves the traditional

k-means algorithm by enriching its input with WordNet hypernyms. The WordNet lexical reference system organizes different linguistic relations into hierarchies/hypernyms (Is-a relation) and W-kmeans uses them as a preprocessing stage before the regular k-means algorithm. We extended this algorithm to the domain of user clustering in Bouras and Tsogkas [6], where we investigated how user clustering alone can affect the recommender's performance.

In this manuscript, we are incorporating n-gram extraction within our keyword extraction mechanism which operates on news articles and explore the effect that this has on the article clustering process of W-kmeans. Moreover, the W-kmeans algorithm is adjusted so that it utilizes n-grams too, besides mere keywords. Furthermore, we are experimenting with different values of the n parameter so as to determine which gives the best results for our clustering approach as well as the target corpus (i.e. news articles from the web).

3 Information flow

In Fig. 1 we present the information flow of the suggested approach. Initially, at its input stage, our system fetches news articles generated by 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.

Text preprocessing is a key process of the system as a whole, and probably as important as the IR processes that follow it. It is applied on the fetched article's content and results to the extraction of the keywords that each article consists of. At this analysis level, we apply some typical dimensionality reduction techniques, which include: stemming, stopword removal and filtering of low frequency words. In addition to the above, we also utilize techniques regarding:

- Feature selection/reduction where we attempt to select a subset of features that are most useful for the IR tasks that follow. This is achieved (a) via POS tagging and noun extraction, and (b) by pruning of words, appearing with low frequency throughout the corpus, which are unlikely to appear in more than a small number of articles.
- Feature generation/extraction where new features are sought for representation. In our case, this is achieved

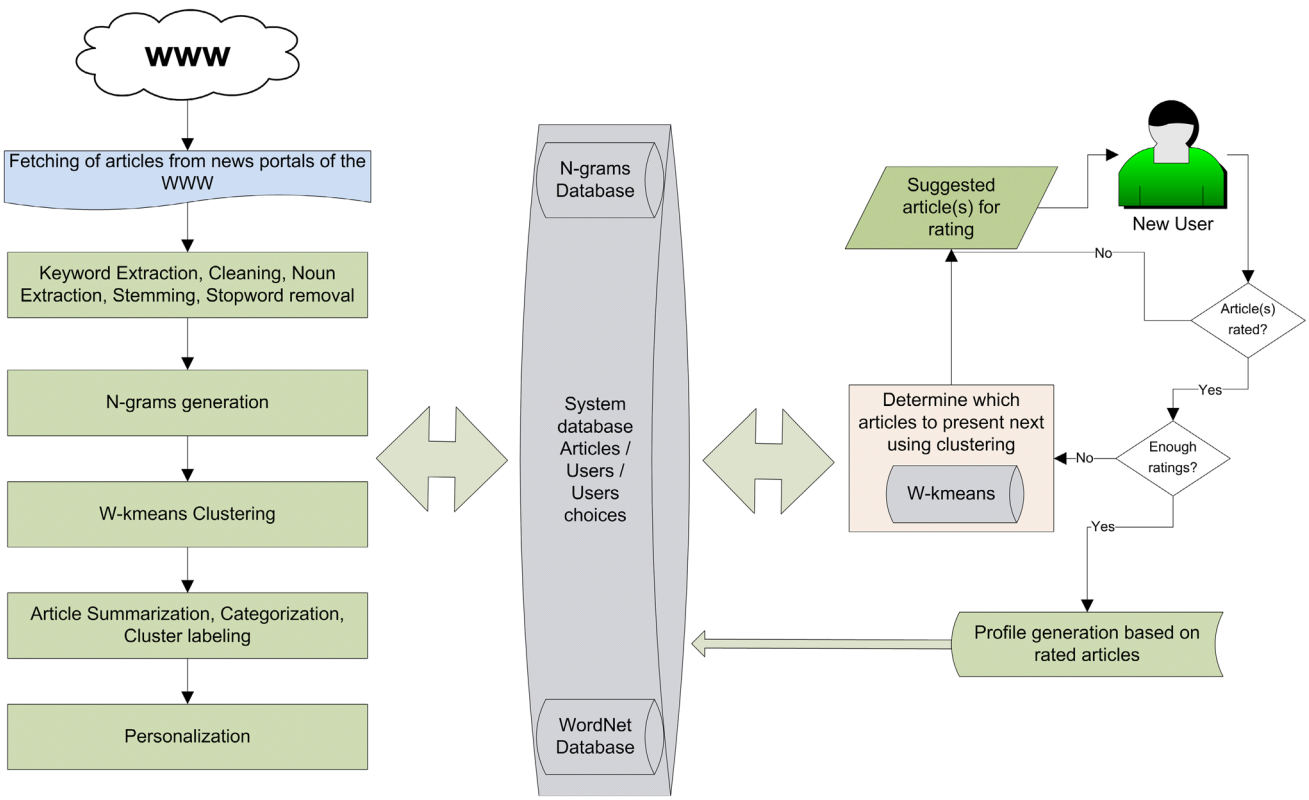


Fig. 1 Information flow of the proposed methodology

in a twofold manner: (a) via the extraction of the text's nouns (through the process of POS tagging) and (b) via WordNet generated hypernyms. In particular, for W-kmeans we enrich the article's text with new words utilizing the WordNet tree-like structure of hypernyms.

Keyword extraction, making use of the vector space model, generates the term-frequency vector which describes each article as a 'bag of words' (words–frequencies vector) to the key information retrieval techniques that follow: article categorization, summarization and clustering. Previously, in Bouras and Tsogkas [5], we had successfully enhanced the efficiency of this 'bag of words' with the use of an external database, WordNet, in order to improve the results of the clustering algorithm. Pruning of words, appearing with low frequency throughout the corpus comes next. These words are unlikely to appear in more than a small number of articles and are thus of no retrieval value. The above characteristics of our system give its content-based nature.

This enhanced feature list feeds the k-means clustering procedure that follows. It is important to note, however, that the clustering process is independent from the rest of the steps, meaning that it can easily be replaced by any other clustering approach.

Following the core IR tasks of our mechanism, the personalization algorithm takes place. The personalization module can easily adapt to subtle user preference changes. Those changes, as expressed by the user's browsing behavior, are detected and continuously adjust 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.

User profiles from multiple users and timeframes are then clustered using the W-kmeans 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. Those profile clusters are used at the recommendation stage in order to enhance the system's usage experience, by providing more adapted results to users revisiting the site. When a user comes back his/her clustered profile is recalled. Articles matching his/her profile are extracted and are considered for user recommendations.

The contribution of the present work is twofold.

- We are enhancing the feature generation process of the preprocessing stage by also extracting n-grams from the text and indexing them into the database. The process is similar to that of keyword extraction (which could actually be thought of as the trivial case of n-gram extraction, with $n = 1$): for each article and for values of n from 2 to 6, we identify the word n-grams of the input text and properly index them into our database. In this scenario, the overall similarity either between two articles or between an article and a class, i.e. cluster, cannot be conveyed only in terms of keyword frequency/inverse document frequency metric (kf-idf), but rather as a combination of kf-idf and its n-gram counterpart metric, let's call it gf-idf, which will be described in detail in Sect. 3. Note also that stemming is not being applied on the extracted n-grams since stemming techniques can be used in word-based systems but not in n-gram-based systems.
- We are targeting the new user problem: when a new user enters the system, he/she enters a priming phase (user-prompting) in which a series of articles are suggested to him/her as rating candidates. At this stage, we want the total number of presented articles to be minimized, while also allowing the system to gain as much information as possible from the rated articles. Determining which articles to select for the user prompting phase and the order in which to select them is the main aspect that the current work focuses on. Initially, we suggest articles for user rating based on a given strategy. Even though one would expect that this strategy would be of significance, this is not really the case as already shown by Rashid et al. [25]. Thus, we simply select articles coming from the most rated list—a list that resides in the database. While the suggested articles are not rated by the user, we continue the suggestions based on this list. Once the user has rated one article, we utilize our clustering data in order to find out and suggest for subsequent rating, articles that (a) belong to the same article cluster as the rated one or (b) are chosen by users that have rated the selected article(s) likewise before. The above procedure continues until enough user ratings are gathered. Once this phase is complete, the user profile has been bootstrapped and the system goes into its normal recommendation state.

Summarizing, in the current work, W-kmeans operates using (a) keywords, (b) enriched hypernyms from keywords and (c) previously extracted n-grams. The effect that these three aspects of the preprocessing steps have on the clustering process will be presented in the upcoming sections.

4 Algorithmic approach

In this section we will be describing the various algorithmic steps that are followed in our approach in order (a) to tackle the new user problem, and (b) to enhance the keyword extraction mechanism using word n-grams.

4.1 Tackling the new user problem

Algorithm 1 presents the procedure that is used for gathering ratings from a new user who is visiting the system for the first time (and registers to it).

Algorithm 1. Determining which articles to present to the user

```
Algorithm harvest_user_ratings
Input: NULL
Output: user_ratings[] //rated articles by the user
   rated_article = NULL // first rated article A1
   article_cluster = NULL
   articles_next [] = NULL
   rated_articles[] = NULL
   while (!rated_article and rated_article < average_rate (article))
      rated_article = rate(present_next_most_rated_article())
      //continue presenting from the L1 list until user has rated 1 article
   user_ratings[] += rated_article // Article A1 is rated with score S1 > average_rate
   article_cluster = find_article_cluster(rated_article)
   articles_next[] = find_most_rated_articles(article_cluster, M)
   // articles_next[] is now list L2 containing M items
   while (has_next(articles_next))
      rated_articles[] = rate(present_next_article(articles_next))
   if (!rated_articles[]) // user hasn't rated any of the M articles from L1
      articles_next[]=find_most_rated_articles_from_user_clusters (rated_article,M)
      // articles_next[] is now list L3
      rated_articles[] = rate(articles_next[])
      user_ratings [] += rated_articles[]
      GOTO: T // Continue possible suggestions from user clustering
  else //user has rated some of the M articles
      user_ratings [] += rated_articles[]
T:    while (user_ratings.size() < Rmin) // do we have enough ratings?
      articles_next[]=find_most_rated_articles_from_user_clusters(rated_articles,M)
      // articles_next[] is now list L4
      rated_articles[] = rate(articles_next[])
      user_ratings [] += rated_articles[]
  return user_ratings[]

Function average_rate
Input: article
// Retrieves the average rate given for this article by any user who has rated it
Output: average rating

Function rate
Input: articles[]
// Presents for rating the selected article(s) and returns the rating scores
// or null if the article wasn't rated
  rated_articles[]=NULL
Output: rated_articles[]

Function find_article_cluster
Input: rated_article
// Recovers the cluster that the article belongs to
Output: article_cluster
```

Algorithm 2 outlines the steps used to recover articles based on either article or user clustering information. We don't get into much detail on how each function works, but the names should be self-explanatory.

Algorithm 2. Recovering articles based on article or user clusters

```
Algorithm find_most_rated_articles
Input: cluster, M
// Recovers the M most rated articles that belong to this cluster
// Uses article clustering results from the database
// cluster can be either an article or a user cluster
Output: articles[M]

Algorithm find_most_rated_articles_from_user_clusters
Input: article / articles[], size M
// Recovers articles from the user clusters which contain users who have
// previously rated the specified article(s). Uses user clustering results
Output: rated_articles[]
   rated_articles[]=NULL
   clusters[]= find_user_clusters(article)
   //find clusters of users who have rated article(s)
   for each cluster in clusters[]
      rated_articles[]+=find_most_rated_articles (cluster, M)
   return rated_articles[]
```

In our work, we use an item by item personalized strategy, as the one described by Rashid et al. [25], in order to select articles as rating candidates for the new user. Initially, when a user enters the prompting phase, we present to him/her articles, one by one, from the most rated (popular) list of articles that reside in the system's database. Let's call this initial list: L_1. The presentation of articles continues until an article, say A_1, is rated by the user with score S_1. If the S_1 score is less than the average score for this article (as rated by other system users), user prompting continues from the L_1 list. Otherwise, if S_1 is above the average score given for A_1, we consider this article as representative of the user interests and, as such, the cluster that this article belongs to is retrieved from the database. Following, we can suggest for user rating M of the most rated articles from this cluster, which are forming a new article list, say L_2. Note that L_2 contains articles based on article clustering information originating from the system's database. Choosing the correct value for M is a matter of experimentation since there is a specific trade-off for it, which we will try to briefly describe. Large M values give many similar articles, i.e. from the specific article's cluster, so that a hit there, regarding the user's preferences, will probably harvest many user ratings. However, if the rated article A_1 does not convey entirely the user's interests many articles that with high probability won't get rated will be presented without an easy way for him/her to backtrack. This can have a negative effect on the system's performance and will probably also cause user frustration. On the other hand, small values of M might lead to a similar negative impact via a different path: a user would expect from the recommendation system to catch up his/her preferences quickly and not backtrack to articles he/she has no interest about. In a nutshell, we don't want to overload

the user with articles from a single cluster, but still, we want to determine relatively fast if articles belonging to that cluster are really interesting to the user. Additionally, we want to cover as broadly as we can the related clusters that might capture user ratings, thus we expect that a small to medium value for M should be more reasonable.

As our algorithm proceeds, if no rating is given on any of those M articles of L_2, we seek for the user clusters which contain users who have previously rated the item A_i with score S_i. Using these user clusters we can form an article list, say L_3, which consists of M * number of clusters of the most rated articles. Again, we choose to keep M articles from each of these user clusters and as before the same tradeoffs apply on the value of M. The L_3 list, containing user clustering information is subsequently suggested to the user and any possible article ratings that are gathered from the user are used to recreate the L_3 list in a similar fashion. This process forms a loop until the total number of ratings reaches our defined threshold, say R_{min}.

On the contrary, if the user has rated at least one of the M articles of the L_2 list, we seek for the user cluster(s) that contain most of the previously rated articles and again, we select the (M * number of clusters) of the most rated articles (list L_4). Although resembling each other, the L_3 and L_4 lists are not the same: the difference lies in that L_3 is based entirely on user clustering, while L_4 is instantiated via article clustering first and enhanced later on via user clustering using collaborative knowledge that resides in the system's database. We have selected this article/user clustering combination based on our previous experimental results in Bouras and Tsogkas [6]. The aforementioned approach continues until at least R_{min} user ratings are gathered.

Once the needed user ratings are harvested, the registration process ends and the user can now browse through the personalized recommendations that the system provides. As explained in Sect. 2, the personalization improves the quality of the user-suggested articles based on the continuous feedback that the user provides via his/her choices.

4.2 n-gram weighting analysis

When keyword extraction completes its operation on each news article that is fetched from the web, a list of stemmed keywords is generated and stored in the database. For example, let's consider a given article that belongs to the domain or cosmology (categorized as 'science' by our classifier) and for which our preprocessing mechanism detects 18 valid keywords, as depicted in Table 1.

In the above list, the existing keywords are stemmed nouns and are presented in decreasing order given their absolute frequency of appearance in the text. Given the above data as well as the recorded data of the rest of the keywords in the database, one could easily determine the tf-idf weights of these keywords.

Table 1 Stemmed keywords with accompanying frequencies as extracted from an article

ID	Keyword	Frequency
1	Year	6
2	Cosm	4
3	Radiat	4
4	Profess	4
5	Mass	3
6	Intens	3
7	Event	3
8	Neuhauser	2
…	…	…
18	Burst	2

Table 2 Top n-grams, with $2 < n < 6$ as extracted from the same article

ID	n-gram	Frequency
1	Light years	4
2	The most	3
3	The past 3,000	3
4	To have	3
5	Light years away	3
…	…	…
24	Professor Neuhauser	2

In addition to the extracted keywords, our approach also extracts n-grams from the articles, with $2 < n <= 6$ and with frequency of appearance $fr > 1$. For example and for the same article, the extracted n-grams are depicted in Table 2.

From the n-gram list of Table 2, one could infer that some n-grams, like: "light years away" and "Professor Neuhauset" could be considered as good representatives of the particular domain that this article belongs to. Intuitively, the cluster that this article belongs to should have those n-grams weights boosted.

Once we have the above vectors, we can come up with a weighting scheme utilizing both the individual keyword and the n-gram information. In this path, we are enhancing the keyword weighing equation described by Bouras et al. [4], where the score of sentence a, S_a is given from equation (1).

$$Sa = \sum w_{k,i}(k_1 + k_2)k_3 * k_4 \quad (1)$$

where k_1 expresses the impact of each particular keyword i that appears in the article's body, given its relative

frequency compared with the total number of appearances in the database (tf-idf), k_2 expresses the impact of keywords appearing also in the article's title and factors k_3 and k_4 express the impact of the categorization and summarization sub processes (as depicted in Fig. 1) respectively (for a more in depth analysis of the factors k_1-k_4, we suggest that the user reads Bouras et al. [4]. The previously described weight was previously also utilized in our W-kmeans algorithm [5].

Following and enriching this weighting notion, we can assign weights to n-grams that are in each text too by utilizing their 'tf-idf' statistics. More specifically, for each n-gram j, its weight is expressed by its tf-idf frequency—let's call it gf-idf (gram frequency/inverse document frequency). This weight could be written as described in equation (2).

$$W_{ngi} = gf - idfi = \text{freq}_j * \log \frac{N}{M} \tag{2}$$

where N is the total number of articles in the database and M is the total number of articles containing the n-gram i.

Combining (1) and (2), we could express the total weight of each sentence, given its keywords and n-grams as in equation (3).

$$\text{Si} = A * (\sum w_{k,i}(k1 + k2)k3 * k4) + B * \sum w_{ngj} \tag{3}$$

We can control and normalize the effect that the keywords and n-grams have on the clustering algorithm in a linear way by using the two parameters A, B, mentioned in (3), so that:

$$W'_{kwi} = W_{kwi} * A \tag{4}$$

$$W'_{ngi} = W_{ngi} * B \tag{5}$$

and:

$$A + B = 1 \tag{6}$$

Determining the appropriate weights A and B is a matter of experimentation for the given dataset. Our analysis, however, focuses only on news articles originating from the web.

Summarizing the weighting heuristics described in this section, we could propose the algorithmic steps given in Algorithm 3 for clustering news articles using n-grams.

Algorithm 3. Clustering news articles based on keywords, enriched hypernyms and n-grams

```
Algorithm clustering_based_on_ngrams
Input: articles, number of clusters, X
Output: cluster assignments
for each article a
KW = fetch 20% most frequent k/ws for a
NG = n_grams_extract(a, 1<n<X)
WN = wordnet_enrich(a)
clusters = kmeans(KW, NG, WN)
// Weighting is on a sentence level, //based on (3)
return clusters
```

5 Experiments and results

For our experimental procedure we built a dataset using a snapshot of our system's database which contained news articles previously crawled from the web. The reason that we didn't select to work with other datasets (like NetFlix or MovieLens) is due to the clustering capabilities of our system: W-kmeans is enabled and already applied on our dataset.

In order to determine the system's efficiency, we employed one of the most widely used evaluation metrics for predicting performance of recommender systems: the Mean absolute error (MAE). We use the MAE metric, for expressing the average absolute deviation between the predicted and the actual user ratings. Mean absolute error can be computed using formula (7).

$$\text{MAE} = \frac{\sum r'(u,i) \in R' \, |r(u,i) - r'(u,i)|}{|R'|} \tag{7}$$

where r(u, i) $\in [1, 5]$ is the actual rating (as recorded in the system's database) of user u for article i and r'(u, i) $\in [1, 5]$ the predicted/recommended preference for user u of articles belonging to the space of proposed articles, R'.

Another criterion we made use of during the system evaluation is the Clustering Index (CI) one, defined as:

$$\text{CI} = \bar{\sigma}^2/(\bar{\sigma} + \bar{\delta}) \tag{8}$$

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, CI focuses on increasing the first measure (intra-cluster similarity) while decreasing the second (inter-cluster similarity).

5.1 Dealing with the new user problem

In order to determine how well the proposed approach regarding the new user problem behaves, we utilized ratings coming from real system users. We firstly eliminated users who had fewer than 50 recorded ratings. This left us with 60 users who had rated in total 2,055 articles with over 10,000 ratings. Using a cutoff of 50 ratings is significant: we want to prevent users who haven't used extensively the system from affecting the evaluation process. In general, we need many ratings for each user in order to have a good sample of articles that they were able to rate which effectively constitutes their preferences. Since a "new" user in the experimentation that follows is practically each one of the 60 users we previously mentioned, for each run we withheld all of the user's ratings from the system, i.e. as we presented articles based on each

of the strategies that we are evaluating, these suggestions were without any a priori knowledge. Intuitively, we define that users have "rated" articles when they have "viewed" them, i.e. they had article ratings in the database. This means that if an article that is presented to the user was found as rated or viewed in our database we consider it as a successful proposal for user rating. Regarding the particular scores, with the rating score range being from 1 to 5 (5 being the 'most liked'), the rating score found in the database is used as the rating score a user would give during the prompting phase. If the article was found in the viewed list, we consider this with the highest rating score, i.e. 5. For our first and second experimentation set, we stopped presenting articles once we got the required number of ratings, which for our experimentation was set to be:

$R_{min} = 20$.

For our first experiment, we tried to determine the best value of the parameter M described in Sect. 3. That is: the best amount of articles we should present to the user, which belong to a certain cluster (either article or user cluster) after one or more article ratings are harvested from the user.

For this experiment we used an increasing number of values for M on each run, starting at $M = 1$ and ending at $M = 50$. The results are presented in Fig. 2.

From the graph of Fig. 2 we can pinpoint that the best value of the parameter M in terms of MAE is $M = 5$. From a natural point of view this means that selecting 5 articles from the articles or user clusters is best for forming the M_2, M_3 or M_4 lists. Executions with lower M values suffered from few article suggestions coming from article and user clusters, something that was leading to low performance and longer user prompting times. We expect that on an online experiment with many users, the performance would have been even worse, counting in also the user frustration which is to be expected when the prompting phase takes too long. Evidently, values for $M > 10$ also suffered from poor performance. This can be explained by the fact that when we are using too many articles from article and user clusters, the users have a hard time backtracking out of a cluster if they need to when going through the prompting phase. For our subsequent experiments, we are using $M = 5$.

For our second experiment we used each of the user prompting strategies described in Sect. 2b: entropy, random, popularity, balanced, personalized item by item, as well as our proposed clustering-based strategy, in order to select which articles to present to the new user. Once the prompting phase completed for each strategy, we calculated the number of articles that the user had to view until *the* $R_{min} = 20$ ratings were harvested. We need to stress out

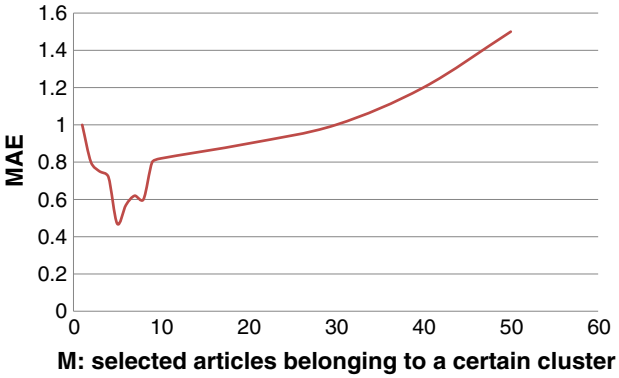


Fig. 2 MAE scores for various numbers of M

that the fewer the articles that we had to present to the user the better, given the fact that we are saving user effort. Also note that for the personalized item by item strategy, we used the popularity based approach for presenting the initial articles until a rating is accomplished by the user. This is similar to the proposed clustering-based methodology that we follow, except, of course, that we are also exploiting the clustering information. After that, we found out which articles are close to the rated one by utilizing the cosine similarity as a measure. The articles' similarity is calculated by using each article's keywords as explained by Bouras et al. [4]. The similar articles are next used as suggestions for users to rate.

From the graph depicted in Fig. 3, we can observe that our clustering-based article selection approach clearly outperforms all other strategies. As explained in Table 3, the random strategy required an average of 136 articles to be presented to the user before 20 ratings were harvested. This result is expected due to the random nature of this strategy and the fact that the user ratings cannot be considered as uniform throughout the dataset: each user is expected to have rated articles that only appealed to him/her and only for particular domains. The same number (for the amount of articles) was 113 in the case of the entropy-based strategy and 70 for the popularity-based strategy. The results for the entropy strategy, though surprising, have a plausible explanation: this strategy promotes less popular articles. However, there is a straight correlation between popular articles and the chance that a new user would like a popular article. Thus, by choosing less popular articles most of the times, this strategy suffers from bad performance. The results were better for the balanced popularity*entropy strategy with 63 articles on average. The personalized item by item strategy, even though very promising with an average of 41 articles, couldn't match the average of 32 articles that our approach scored.

For our third experiment, we tried to determine the prediction accuracy of the proposed approach compared to

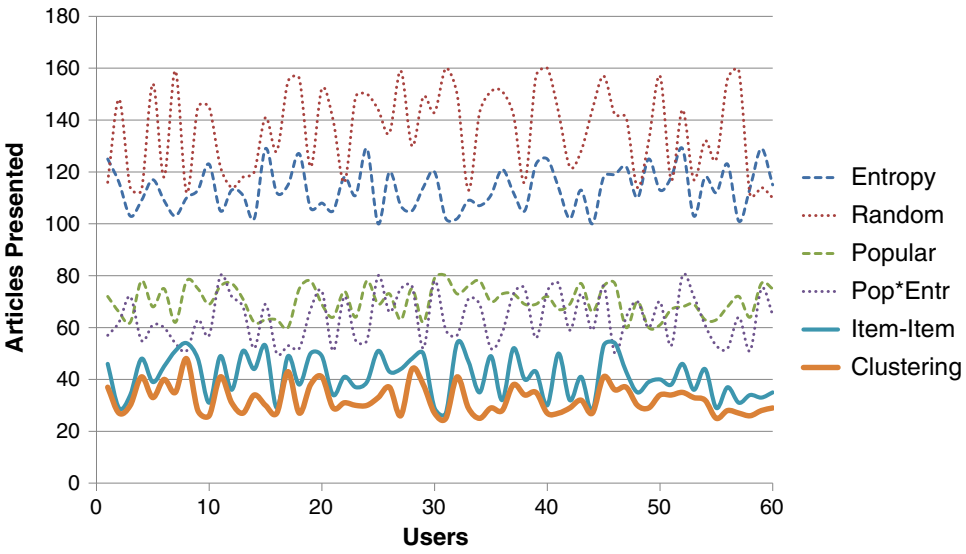


Fig. 3 Required number of articles per strategy during the prompting phase

Table 3 Average required number of articles per strategy until Rmin = 20 is achieved (the less—the better)

Strategy	Average articles
Entropy	136.3
Random	113.3
Popularity	70
Balanced	63.6
Personalized	41.3
Clustering	32.3

the previously mentioned strategies. Again, we made use of the MAE metric. For determining the MAE scores of each strategy, we presented for user rating a total of 30, 50, 70 and 90 articles in four subsequent runs for each of the six strategies. The results, presented in Fig. 4, show the MAE variations of the different strategies as a function of the presented articles.

We can observe a MAE improvement as the number of presented articles increases, something that has a significant effect especially for the proposed clustering-based article selection strategy: as more and more articles are rated by the user, our approach can pick up better candidates for user rating by utilizing article and user clustering data from the database. Indeed the proposed approach gives the lowest MAE scores for each of the experiment's executions. Another result we can pick up from Fig. 4 is that the random strategy has the worst prediction accuracy, validating our observations on the first experimentation. We can also observe that the personalized item by item strategy is again, as in the second experiment, close to our proposed strategy.

5.2 Evaluation of n-gram enrichment

For our second part of the system's evaluation, we tried to determine the effect of n-grams extraction and usage within the context of news article clustering and, in particular, onto the efficiency of the W-kmeans algorithm via a series of offline experiments. For extracting the n-grams from the news article's body, we utilized the n-gram extraction toolset from Zhang (2013) [30]applying it on the documents of the used corpus.

Our corpus consists of 10,000 news articles obtained from major news portals like BBC, CNN, etc. over a period of 5 months. Those articles were evenly shared among the eight base categories featured by our system in order to avoid any bias towards a specific class.

Firstly, we tried to determine the best value for n, i.e. up to what size should the words window be when capturing grams. For this experiment, we arbitrarily set A = B = 0.5 (i.e. giving the same weighting significance to both keywords and n-grams) and tried different values of n where 2 <=n <=6 as shown in Fig. 5. For each n value, we repeated the clustering process 10 times with different starting cluster centers (10-pass experiment).

The results depicted in Fig. 5, show that when n = 3, i.e. we keep both 2- and 3-g for weighting the sentences, the W-kmeans algorithm's performance is increased by an average of 0.3 regarding the clustering index of the generated clusters. This result is in accordance with the findings of Furnkranz [12]. For n = 4, we still see a performance increase compared to the case where n-grams are not taken into consideration (i.e. n = 1 in the graph of Fig. 5).

Increasing even further the window size seems to have a negative impact on the overall clustering index results.

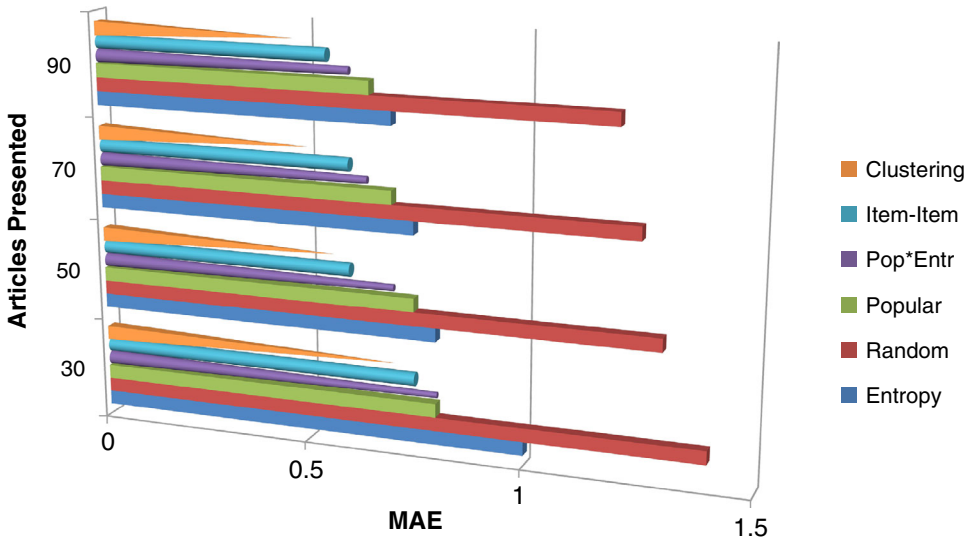


Fig. 4 MAE scores for each of the prompting strategies for various number of presented articles

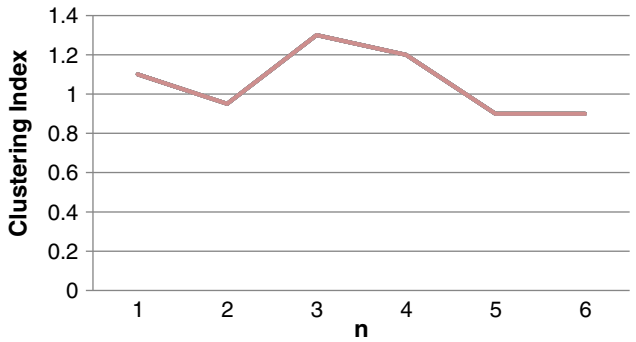


Fig. 5 The effect of the parameter n when considering n-grams for W-kmeans clustering

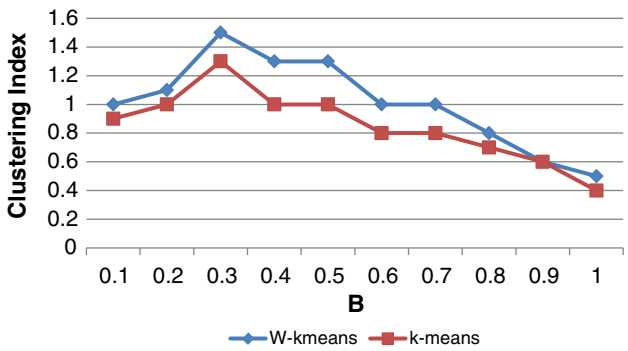


Fig. 6 Performance results for various B values—comparison of W-kmeans and regular k-means

This can be explained as follows: larger window sizes mean that n-grams that randomly appear together in larger sequences are weighted much more than they should, a situation that probably has a negative impact on the overall weighting, given their random nature. Another interesting finding of this experiment is that for n = 2, the results get slightly worse than when not using n-grams at all. A possible cause for that might be statistical anomalies in the underlying data, or even poor extraction for n-grams given the library that we used. Extracting, thus, in some cases 2-g that are not really good candidates has less to the observed result, which was later repaired when using both 2 and 3-grams for the weighting process.

In our subsequent experiment, we tried to determine the best values for the keyword and n-gram weighting factors A and B. As such, we set n = 3 per our previous result, and run a 10 pass of our W-kmeans algorithm as well as of the standard k-means algorithm, with increasing values of B (given that from (6): A = 1−B), while recording the CI scores of all the clustering passes. The average CI results are depicted in Fig. 6.

As illustrated in Fig. 6, the best CI results are obtained when B = 0.3, i.e. when the n-gram weighting contributes by a factor of 30 %, while the rest is contributed by regular BOW weighting. What's more interesting is that the performance deteriorates quickly as B increases and reaches the worst value of around 0.5 when only n-grams are taken into consideration.

This can be explained by the fact that not all the articles have outstanding and/or frequent n-grams, and as such, the more we take into consideration n-grams more than regular keywords, the less the performance will get on average. For example, a situation that an article does not have n-grams with frequency of at least 2 is a common scenario for small articles.

Another observation we can easily make, is that W-kmeans outperforms regular k-means significantly even when n-grams are thrown into the weighting equation. As a matter of fact, the results were consistently in favor on W-kmeans for any value of B that we tried. This is a good indication that the WordNet heuristic that W-kmeans applies pays off at any configuration. Nevertheless, we

could not explain the almost identical CI results we got on average when B = 0.9. Our point of view is that this is due to the underlying data and statistic anomalies of the used articles.

6 Conclusions

In this manuscript we have presented a personalized strategy for dealing with the new user problem via prompting in the domain of news articles. We implemented and evaluated an algorithmic approach that makes use of article and user clustering information, in order to decide upon which articles to present next for rating during the process of user registration. Our approach has similarities to the personalized one by one strategy, but the results have shown that it performs better than any of the most commonly proposed strategies for the problem of user prompting.

Our experimentation, based on offline system user ratings, showed that by using M = 5 articles from each article or user cluster we get the lowest MAE scores. Using this result, we later determined that our approach requires, on average, 32.3 articles to be suggested for user rating in order to acquire 20 ratings; an amount of articles that was considerably less than any other techniques that similar systems use. Finally, we compared the MAE scores of the proposed technique with each of the entropy, random, popularity, balanced and personalized item by item strategies. Each experiment, executed over various amounts of articles has also shown that our methodology outperforms all of the afore-mentioned strategies.

Moreover, we analyzed the implications of word n-gram extraction and use within the scope of our WordNet-enabled clustering algorithm, W-kmeans. We presented a means of utilizing both the classical BOW representation and the n-gram expansion in a tunable fashion, via the parameters A and B.

Our experimentation towards determining the best value for the n parameter revealed similar results with the existing literature, i.e. using 2- and 3-g for the weighting process seems to yield better performance in terms of CI when it comes to clustering news articles from the web. Furthermore, we saw that it is not enough to simply include n-grams into the equation. The weighting, given to the n-grams compared to regular keywords, as conveyed by the parameters A and B, is thus of great importance, not previously explored and an area that we feel is domain-specific and open for further experimentation. To summarize, the best results were obtained when n-grams affected the weighting process by around 30 % and this was true for both the W-kmeans and the regular k-means clustering algorithms. Extending on the experimental results of our previous work [6], we again came to the conclusion that W-kmeans outperforms regular k-kmeans, even with the n-gram enrichment process that this work describes in place.

7 Future work

Even though the above results are encouraging, they cannot be considered as conclusive, since they were done over an offline snapshot of the system's database based on already recorder user ratings. Thus, for the future we are considering a more large scale experimentation that will also include online data by more users registering through our system. We would also like to evaluate our approach with other databases too, such as the NetFlix and MovieLens datasets. Furthermore, we are working on enhancing our user prompting algorithm so that it includes more personalized information that can be harvested from the system. In particular, we are looking for an extension that will include the categorization as well as the summarization information which will enhance the personalization factor of our algorithm.

In addition, we are planning on enriching the various components of our system with various and improved techniques, particularly for keyword extraction/enrichment and categorization. We would also like to execute a larger experiment for validating the presented in this work results; with a wider range and randomly selected news articles, we should be able to pin down the best values for n and B regarding the domain of clustering news articles from the web. We will also be focusing on creating suitable communication channels for delivering the article recommendations.

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. Abou-Assaleh T, Cercone N, Keselj V, Sweidan R (2004) Detection of new malicious code using n-grams signatures. Second annual conference on privacy, security and trust. Fredericton, NB, pp 193–196
2. Barron-Cedeno A, Rosso P (2009) On automatic plagiarism detection based on n-grams comparisons. In: Proceedings of the European conference on information retrieval, ECIR-2009, pp 696–700
3. Balabanovie M, Shoham Y (1997) Fab: content-based collaborative recommendation. Commun ACM 40:66–72
4. Bouras C, Poulopoulos V, Tsogkas V (2008) PeRSSonal's core functionality evaluation: enhancing text labeling through personalized summaries. Data Knowl Eng J 64(1):330–345 Elsevier Science

5. Bouras C, Tsogkas V, (2010) W-kmeans: clustering news articles using wordnet. In: Proceedings of KES vol. 3, pp. 379–388
6. Bouras C, Tsogkas V (2011) Clustering user preferences using W-kmeans. In: Proceedings of the seventh international conference on signal-image technology and internet-based systems (SITIS), pp. 75–82
7. Cavnar W, Trenkle J (1994) N-gram-based text categorization. In: Proceedings of SDAIR-94
8. Cleary J, Bell T, Witten I (1990) Text Compression, Prentice Hall
9. Crane M (2011) The new user problem in collaborative filtering. Thesis for the degree of Master of Science, Department of Computer Science, University of Otago
10. Damerau F, Apte C, Weiss S (1994) Toward language independent automated learning of text categorization models. In: Proceedings SIGIR-94
11. Ekstrand M.D, Riedl J.T,.Konstan J.A (2011). In: Collaborative filtering recommender systems, Found. Trends Hum. Comput. Interact 4
12. Furnkranz J (1998) A study using n-grams features for text categorization. Technical Report OEFAI-TR-98-30, Austrian research institute for artificial intelligence
13. Golbandi N, Koren Y, Lempel R (2010) On bootstrapping recommender systems. In: Proceedings of the 19th ACM International Conference of Information and Knowledge Management, ACM, pp. 1805–1808
14. Golbandi N Koren Y, Lempel R (2011) Adaptive bootstrapping of recommender systems using decision trees. In: Proceedings of the forth acm international conference on web search and data mining, pp. 595–604
15. Good N, Schafer J. B, Konstan J. A, Borchers A. Sarwar B. J, Herlocker, Riedl J (1990) Combining collaborative filtering with personal agents for better recommendations. In: Proccedings of the 16th international conference on artificial intelligence and the 11th innovative applications of artificial intelligence conference innovative applications of artificial intelligence, Orlando, Florida, United States, pp.439–446
16. Jung K.–Y, Park D., Lee J (2004) Hybrid collaborative filtering and content-based filtering for improved recommender system. Computational Science-ICCS, pp. 295–302
17. Jurafsky D, James H. M (2001) Speech and language processing. Prentice-Hall, Inc, 2000
18. Kohrs A, Merialdo B (2001) Improving collaborative filtering for new-users by smart object selection. In: Proceedings of international conference on media features, international conference on media futures, Florence, Italy
19. Koren Y, Bell R. M (2011) Advances in collaborative filtering. Recommender Systems Handbook, pages 145–186
20. Mahoui M, Witten I, Bray Z, Teahan W (1999) Text mining: a new frontier for lossless compression. In: Proceedings of the IEEE Data Compression Conference (DCC)
21. Nguyen H, Haddawy P (1998) The decision-theoretic video advisor. AAAI-98. Workshop on recommender systems, Madison, pp 77–80
22. Park S, Pennock D, Madani O, Good N. DeCoste D (2006) Naïve filterbots for robust cold-start recommendations. In: Proceedings of the 12th ACM SIGKDD International conference on knowledge discovery and data mining, ACM, pp. 699–705
23. Pilaszy I. Tikk D (2009) Recommending new movies: even a few ratings are more valuable than metadata. In: Proceedings of the third acm conference on recommender systems, pp. 93–100
24. Rana S, Jasola S, Kumar R (2013) A boundary restricted adaptive particle swarm optimization for data clustering. Int J Mach Learn Cybernet 4(4):391–400
25. Rashid AM, Istvan A, Cosley D, Lam SK, McNee SM, Konstan JA, Riedl J (2002) Getting to know you: learning new user preferences in recommender systems. Proceedings of the 7th international conference on Intelligent user interfaces. California, San Francisco, pp 127–134
26. Rashid AM, Karypis G, Riedl J (2008) Learning preferences of new users in recommender systems: an information theoretic approach. ACM SIGKDD Explor Newsl 10(2):90–100
27. Sarma TH, Viswanath P, Reddy BE (2013) A hybrid approach to speed-up the k-means clustering method. Int J Mach Learn Cybernet 4(2):107–117
28. Wang X, Wang Y, Wang L (2004) Improving fuzzy c-means clustering based on feature-weight learning. Pattern Recognit Letters 25(10):1123–1132
29. Yeung DS, Wang XZ (2002) Improving performance of similarity-based clustering by feature weight learning. Pattern Anal Mach Intell, IEEE Transactions on 24(4):556–561
30. Zhang L (2013) N-Gram Extraction Tools, http://homepages.inf.ed.ac.uk/lzhang10/ngram.html
31. Zhao Y, Karypi G (2004) Empirical and theoretical comparisons of selected criterion functions for document clustering. Mach Learn 55(3):311–331