DOI 10.2298/CSIS110404070B

Effective Hierarchical Vector-based News Representation for Personalized Recommendation

Mária Bieliková, Michal Kompan, and Dušan Zeleník

Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3,
842 16 Bratislava, Slovakia
{bielik, kompan, zelenik}@fiit.stuba.sk

Abstract. With amount of information on the web, users often require functionality able to filter the content according to their preferences. To solve the problem of overwhelmed users we propose a content-based recommender. Our method for the personalized recommendation is dedicated to the domain of news on the Web. We propose an effective representation of news and a user model which are used to recommend dynamically changing large number of text documents. We work with the vector representation of the news and hierarchical representation of similarities among items. Our representation is designed with aim to effectively estimate user needs and generate personalized list of items in information space. This approach is unique thanks its low complexity and ability to work in real-time with no visible delay for the user. To evaluate our approach we experimented with real information space of largest Slovak newspaper and simulated recommending.

Keywords: news, recommendation, hierarchical similarity, vector-based content representation, user model.

1. Introduction

Web has become perhaps the most important source of knowledge since day by day more and more information is available in the Web. But within the amounts of webpages, documents, pictures, videos or music we often miss what we really need. The amount of news which is published every hour becomes increasingly overwhelming. Furthermore, not only in the case of news, we often find duplicity or several information sources covering the same topics. The users are sensitive to the recency of information and their interests are also changing over time along with the content of the Web.

The logical consequence of such insufficiency is personalization of the information provided on the Web. One basic technique for personalization is the sorting information in the Web or in specific domains, to somehow improve user experience. This requires understanding of user needs and preferences and thus user modeling, which enables us to process information, documents and even multimedia content on the web in new ways and with greater accuracy.

Moreover, there is another motivation which leads more and more companies to adapt to user needs. Overwhelmed customers are less likely to buy what they really want when they are not able to find it. On the other hand, everyone who takes interest in offers which are directly adapted to their needs is a potential customer. Suggesting products, videos, music or news has become important for web users but also for web service providers. Recommender systems have been designed and deployed on the Web to improve user comfort and increase profits. But there are still many areas where to improve recommender systems [1].

The problem with recommending items on the Web lies in searching for a combination of users and items. We often discover user interests by monitoring user behavior, i.e. what users search, what they comment on or what they have already purchased. Item recommendation includes two groups of approaches - collaborative recommendation and content based recommendation.

The idea of collaborative recommendation is in discovering similarities among users and subsequently recommending items according to similar user preferences. This approach uses similar users to search for items which have been displayed and could be interesting for the current user. The content based approach, which we focus on, discovers similarity between items based on their description and recommends items similar to those which have been already accepted by a given user (e.g., bought or viewed).

Both approaches introduce new challenges that must be addressed in practical applications, where a combination of both approaches is most often used to address particular problems of single approaches.

In this article, we present a content-based approach for recommending news, which is a very interesting topic since there are many readers who look for interesting information and want to read news comfortably. In every moment there are approximately 20 thousand active users. In combination with around 250 new articles each day we work with nontrivial real data. It makes it effectively impossible for every reader to read all new articles or search for interesting ones. We address this challenge by designing a recommender system in the news articles domain which reduces reader effort during search for articles covering specific topics.

We work with text documents and employ automatic extraction of keywords, named entities and other important terms. Text processing and similarity calculation are discussed in section 3. Our method is a combination of two separate approaches which were experimentally evaluated on the same dataset. First, we focus on similarity between text documents and combine it with another approach where we focus on the representation of document similarity and a user model. We describe our method for content-based recommendation in section 4 and present the evaluation of our work in section 5.

2. Related Work

Several approaches for recommendation and filtering have been proposed since early nineties. Two basic concepts are often mixed together to bring better results [5], [20]. Collaborative recommendation exploits social elements, when users are grouped into clusters based on their previous activity (preferences, habits, etc.). Recommendation is based on the assumption that items liked by other similar users are also potentially interesting for the current user (Fig. 1). This is also the main drawback of collaborative recommendation, as items not rated by a critical amount of users cannot be recommended. This is critical for high dynamic domains such as news recommending as with frequent changes of the information space recommended items can be obsolete before it achieves necessary popularity to be recommended. Despite this drawback, collaborative recommendation is probably the most used approach today [6,26].

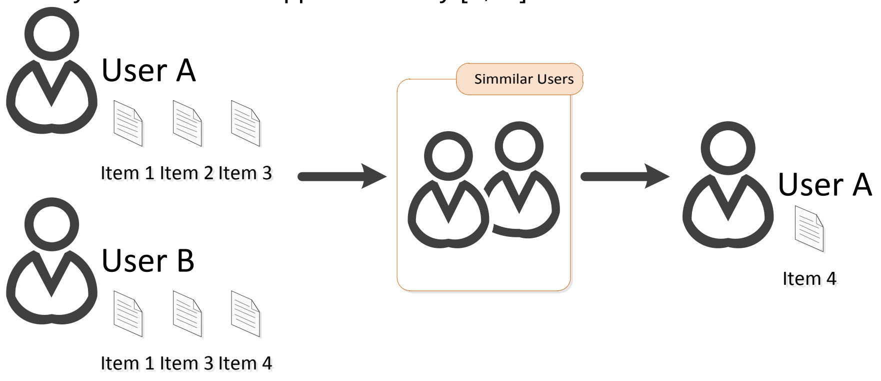

Fig. 1. Collaborative recommendation approach [26]. Users are grouped based on their similarity (visited sites, age, hobbies etc.). We recommend Item 4 to the *User A* because it is the only one item liked by other similar users and not visited by *User A*.

The main goal of content-based based personalization is to identify similar items – to create "clusters" of items instead of users based on associated features (attributes) of the processed items. Recommendation returns similar items to the items positively ranked before (Fig. 2). This type of recommendation is successful in well-structured domains like movies, news etc. [2]. One drawback of content-based personalization is the need for effective and sufficiently expressive item representation as effective similarity computation plays a crucial role (often also being highly domain dependent).
News recommendation and filtering is presently a current research topic with focus on two types of word similarity algorithms: statistical measures based on corpus and semantic distance based on hierarchical organization [30]. These can be further extended by paraphrase identification, vector approximation [31]. Standard methods and their extensions are widely used including n-grams, longest common subsequence, measuring shared syntax or text "fingerprints" [19], [23].

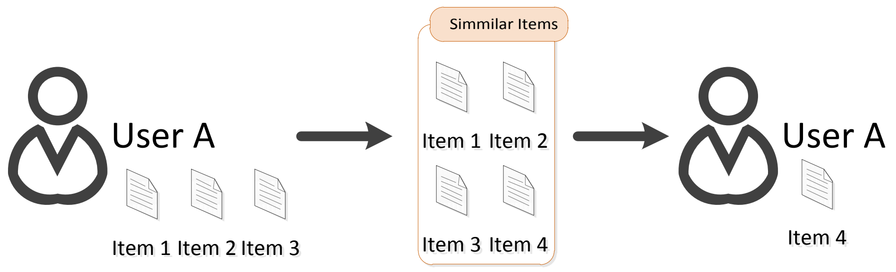

Fig. 2. Content-based recommendation approach. Items are grouped base the content similarity. Item 4 is recommended to the User *A* because of it is the only one item not visited in the group of liked similar items.

A lot of approaches use semantic nearness of documents [22], often based on the WordNet dictionary [32], what is a significant problem when non-English content is processed and fast computation is needed. Various metrics for similarity computation such as Cosine similarity, Euclidean distance, Jaccard Index and many others have been proposed [4].

Text similarity computation in news article domain is however not a developed area. There are various projects in the field of text summarization [9], text classification [24] or categorization [21], latent semantic analysis or SOM [32]. Vector representation of text is widely used in many approaches and by several systems [15], where it is not used for hierarchical or weighting purposes.

The definition of similarity in news recommendation systems has so far been somewhat unclear. For example, similarity can be defined based on news content (similar to plagiarism), or based on the “topic” of a news article, and its definition is extremely important when recommendation lists are created [36].

There are several content-based news recommendation systems. The OTS system [34] provides content-based and collaborative personalization based on association rules and a user interest table. The system works offline because of the large amount of processed data, and users have to choose interesting articles manually.

PURE [35] is designed to recommend medicine articles from a repository to which about 1000 new articles are added daily. Users have to create their own profile by defining interesting articles and the system then recommends new articles based on a classification taxonomy and Expectation-Maximization algorithm once per day.

NewsMe [33] is an adaptive recommendation system based on an open user model, which monitors 81 RSS channels from 21 sources. The core of recommendation method is based on the Nearest Neighbor algorithm. Brusilovsky has shown that a manually and explicitly populated open user model usually results in worse recommendation results in the news domain. Other systems also work with user location to provide location based recommendation [14].

GoogleNews handles thousands of users and articles per day. Three basic approaches are used for recommendation generation: MinHash clustering, Probabilistic latent semantic indexing and co-visitation counts [8], which have all been adapted to the Map-Reduce architecture employed by Google.

Daily Learner [2] is designed to provide adaptive news access based on implicit and explicit user feedback via standard a web interface and also via mobile devices. The user model consists of long-term and short-term user interests and uses k-NN and Naïve Bayes for recommendation.

Content-based approaches suffer from computation complexity, which can be addressed by adding more computational power or finding ways to reduce it. This is important with respect to article information value, which often decreases rapidly. Overspecialization is usually omitted. Recommendation methods should consider returning different numbers of articles for topics, when for example only football is recommended, and also work towards recommendation variability (i.e., prevent too homogenous recommendations) as users likely need not read 10 articles on a topic.

3. Representing News

When recommending news, based on their content, it is necessary to focus on the article content representation effectiveness, because of its high complexity. Besides effective article content representation, effective similarity representation and computation is also needed.

Two aspects must be considered - we try to maximize the useful information extracted from article content, and also need to minimize the amount of processed data (article words). These needs are combined in order to compute fast and accurate content similarity, which is used in personalized recommendation.

3.1. Construction of the Representative Article Vector

The main limiting factor for fast similarity estimation is compact and high precision article vector representation. We propose a vector, which consists of 5 basic parts described in Table 1.

Table 1. Vector representation of a news article.

Title	TF of title words in the content	Keywords	Category	Names/ Places

Title
Article vector comprises lemmatized words from the article's title. It consists of approximately 5 words (based on a 150,000 Slovak article dataset). We

estimate that the title should be a good describing attribute in most occurrences.

Term frequency of title words in the content
We use term frequency to estimate the confidence. If the title is abstract and does not correspond to article content (misleading titles etc.), we easily discover this situation. Term frequency is computed as:

$$tf_i = \frac{n_i}{\sum_k n_k} \tag{1}$$

where tf_i is the term frequency for term *i* (a term from the title) and n_i is the number of occurrences of term *i* in the document (article content) and $\sum_k n_k$ is the sum of occurrences of all terms in the document.

Keywords
Keywords consist of the 10 most relevant keywords for an article. Although many news portals store a list of keywords for every article, these are unfortunately usually at different abstraction level in different portals. This disadvantage can be solved by introducing a custom keywords list, which can be obtained via TF-IDF list calculated over the dataset (100,000 Slovak news articles from the news portal SME.SK). We can reduce the high dimensionality by removing any words except nouns and names. The keywords extraction approach can be easily replaced by any keywords extraction service, while the time consumption issue should be considered.

Category
We include a “tree-based” category vector with weights. This vector is constructed based on a portal specific category hierarchy (optional). The category is important for similarity search, when articles from one category are evaluated as more similar. The weight for every category is estimated as:

```
n=1
For i=|Category| downto 0 do
  weighti=1/n
  n=n*2
end
```

For example, let us consider three articles *A*, *B*, *C* and four categories *C1*, *C2*, *C3*, and *C4*, where the article *A* and *B* correspond to categories *C1*, *C2*, *C3* and article *C* to categories *C1*, *C2*, *C4*. Then the vector for every article is represented in Table 2 and Fig. 3. As we can see, articles *A* and *B* would be more similar then the article *C*. Values in the Table 2 are computed using proposed algorithm.

Table 2. Article Category Vector Representation.

	C1	C2	C3	C4
A	1/4	1/2	1	-
B	1/4	1/2	1	-
C	1/4	1/2	-	1

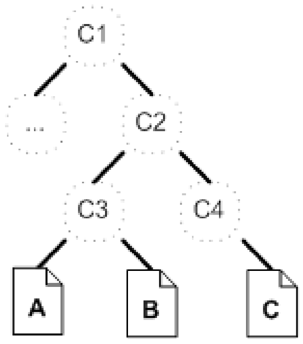

Fig. 3. Example hierarchy of news portal categories.

Names/Places
We include names and places extracted from article content. We extract names or place during the preprocessing stage since our method does not remove full-stops or uppercase letters. We identify names or places as words starting with an upper-case letter without a full-stop before it in the text stream (precision=0.934, recall=0.863). Alternatively, other name extractor systems for English language can be used [10].

3.2. Similarity Computation

For similarity calculation we employ cosine similarity and Jaccard index computation, which is widely used in information retrieval tasks [29]. The similarity of two articles *A*, *B* is computed as:

$$Jaccard\ index = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

Jaccard index was chosen based on our experiments. We used two datasets. The first were articles from news portal (1000 articles), which had assigned (by author) at last one similar article. The second was manually annotated dataset (100 articles). The expert assigned the similarity level (1-5) for each pair.

We computed list of similar articles for every article in the dataset and compared these to the similarity assigned by authors or experts. We calculated precision, recall and F-Score for every dataset and proposed content representation. Results were compared [18] to standard text mining method TF-IDF as shown in Table 3.

Table 3. The Similarity metrics comparison.

Dataset	SME.SK		Manually annotated dataset		
Method	Proposed method	TF-IDF	Proposed method		TF-IDF
			Cosine similarity	Jaccard index	
Precision	0.165	0.091	0.700	0.843	0.511
Recall	0.202	0.117	0.816	0.818	0.587
F-Measure	0.182	0.102	0.753	0.870	0.546

As we can see there is a huge increase of precision and recall using proposed content representation. Poor results for the real “similar” dataset obtained directly from news portal can be explained by not accurate and incomplete information on the news portal. An author of a new article chooses none, one or two similar articles intuitively nowadays, and our method found probably more similar articles. Based on these results we decided to use Jaccard index because of the better results – computation complexity and F-Measure.

Every part of the article vector has its own global weight. By changing this weight we adjust the calculated similarity and its precision. Since we calculate article similarity online in a fast way, we can dynamically change these weights to obtain new results. For example if a user reads all recommended articles, we can change the weight of the category part to zero and recalculate similarity to obtain a new set of similar articles from different categories.

4. Personalized News Recommendation

Based on the proposed approach to article representation, we compute similarity between articles. To personalize news we work with large numbers of articles which are possibly interesting to individual users. We have devised a method to efficiently model user preferences and recommend interesting articles via hierarchical representation based on the aforementioned vector representation of articles and similarity computation. We designed this representation to support incremental addition of new articles and enable real time computation of article similarity.

4.1. Tree Representation to Store News

The main aim of our representation is to group articles using their similarity. A typical usage scenario is the retrieval of a list of similar articles to a selected article. Using a similarity matrix is unfeasible due to its high memory complexity and has difficulty of real-time recommendation with the large number of articles in our dataset (we have more than 250 new news articles daily).

We use binary tree to represent similarity relations. Our motivation to use tree structure is low time complexity of storing and retrieving items and its ability to be created incrementally. In our approach we follow the simple concept where real articles act as leaf nodes. The hierarchy itself is generated over these articles and is used for representation of metadata to access similar articles (see Fig. 4).

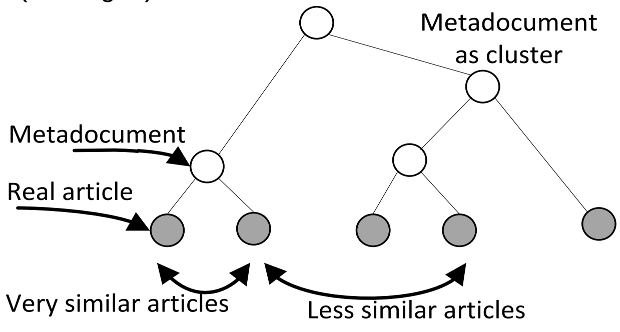

Fig. 4. The hierarchy is built over real articles. Connections are made based on similarity between articles. Closer articles are more similar than articles which are further apart. Metadocuments acts as clusters created by aggregating their child nodes.

Our relations hierarchy is incrementally built similarly to the hierarchy presented by Sahoo [25]. In our hierarchy, we rely on a repository which contains current articles and also assume that they are properly organized. A set of words extracted from articles and normalized is used as features to compute similarity between articles. Each node in the tree is labeled by a set of features, while edges in the tree represent the hierarchy which keeps similar articles nearby. We designed our representation as a hierarchy where:

- real articles are placed at the lowest level of the tree as leaf nodes,
- features are spread to upper levels of the hierarchy structure,
- similarity is stored in the hierarchy itself.

The hierarchy is built incrementally, when an article is downloaded as soon as it was published and added to the hierarchy structure. We use special nodes (metadocuments) to aggregate information (union of features) stored in the children documents. Metadocument itself is also represented as proposed vector of features used for real articles (title words, keywords, category, names and places). This is important especially for applications where real-time retrieval is needed (e.g., recommenders). With every new article we modify the tree to satisfy the rule that more similar articles are closer in its structure. Fetching similar articles is then only an issue of locating nearby nodes. The built hierarchy is large, but on the other hand it is compensated by the fast storage and retrieval processes.

When a new article is processed and ready to be added to the structure we:

- locate a place in the tree where to add the article,
- add the article at the correct place and adjust edges in the hierarchy,
- modify the rest of the structure which is affected by the new article.

Searching for the best position for a new article starts at the root of the tree. We proceed step by step through each node were every node has assigned vector representation which describe real articles occurring in the corresponding subtree. The decision on placing the article is made using the Jaccard's similarity (higher similarity wins). Tree traversal ends once a leaf node is reached or the calculated similarity is almost equal for each branch.

To insert a new article we split the branch at a designated place, where the article should be inserted, and append a new node for the new article. We also modify the parents of the new node by spreading information about the new article and aggregating features as needed (i.e., unify features as shown in Fig. 5).

We use all of the features extracted from news articles. To prevent too many of the features at the top of the tree (root), we discard selected features (features with low TF-IDF) when they are spread to the root. We presume that every feature is relevant, but features which are rare do not affect the decision process at the beginning of search. We reduce these rare features because their weight is low. We calculate the weight as the ratio of feature occurrence and occurrence of other features. Rare features are spread only if they reach a weight comparable to other features which is more likely to happen deeper in the tree.

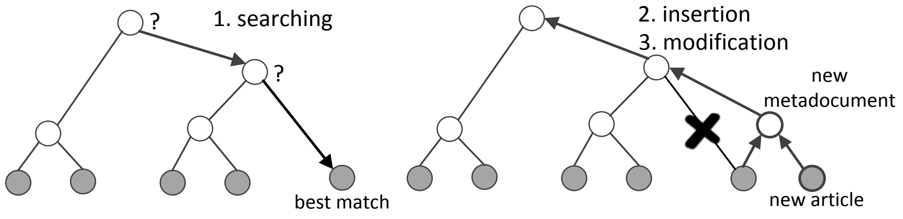

Fig. 5. In the first step we find the best place to insert a new article (left picture), which is added at the correct position in the second step. Metadata are created for the new node (right picture) and features are propagated to the root.

4.2. Modeling User Preferences Using Tree Representation

We discover interests of individual users by monitoring the activity of each reader. Similarly to the work by Carvalho et al. [7] we analyze records of users' activity. Articles that readers view are located in the hierarchical structure we described above, which contains relations between similar articles. We discover user interests by using the records of user activity and the hierarchy. During recommendation we constrain the number of recommended articles to a limited number in order to prevent information overload. We also seek equilibrium between recency and relevancy to maximize recommendation precision.

A prerequisite for our method is that each individual has some interests, what can be easily verified using the history of article views for particular readers and evaluating their interest in certain categories or sections of the news portal.

Similarly, there are identifiable fields of interests for each reader, for who it makes sense to explore other interests based on the calculated similarity between articles. We substitute this metadata (categories) created by editors with the hierarchy of similarity relations which provides our own metadata.

There are several options how to calculate similarity based on the content itself. There are also sophisticated methods, which are able to determine semantic similarity [11]. However, simple text similarity is often used in news recommendation with good results [17]. We use Jaccard's similarity to calculate article similarity with the aforementioned customization of vector-based article representation.

Figure 6 shows our method for discovering of reader interests. Here we present our understanding of user profile [13] in the hierarchical form. We use the tree structure created using the similarity of articles and records of user activity. We have a hierarchy of nodes which effectively represents similarity of real articles even without actual similarity calculation between particular pairs. Thick edges are paths from the displayed article to the root of the tree. Nodes where thick edges merge denote fields of interests.

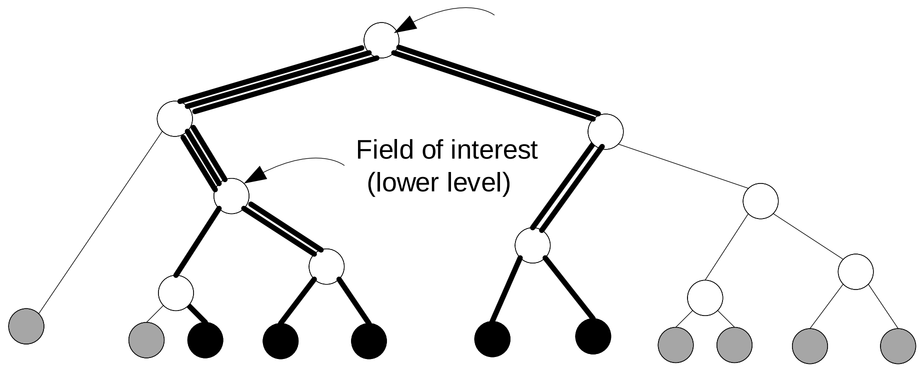

Fig. 6. User interests originate from black nodes representing already displayed articles. Bold lines connect displayed articles with the root node. Multiple lines represents the intensity of user interest in specific cluster of articles.

In this way, we discover interests for each user using their reading history. One user interest corresponds to one node in the tree which is used to define a hierarchical set of articles belonging to this interest (articles in the subtree).

When considering a reader's fields of interests we compare granularity of the interests depending on the depth where the node in the tree is located. Fields of interest that are closer to the leaves of the tree are more focused on a particular topic (e.g., articles about hockey). Fields of interest that are closer to the root correspond to more general topics (e.g., articles about sport).

4.3. Recommendation Generation

Recommending specific articles using our proposed hierarchical structure is a matter of selecting articles based on several reader interests. We first find the relevant interests for a particular reader, and calculate the relevance of the interest as the ratio of articles displayed from this interest and all articles belonging to the interest. Thus, we are able to sort interests and prepare for the selection of appropriate articles. Figure 7 shows the selection of interesting articles for a specific reader. Highlighted articles, which the user has read, are used to determine the relevance of his interest, while other articles are potentially interesting for the reader.

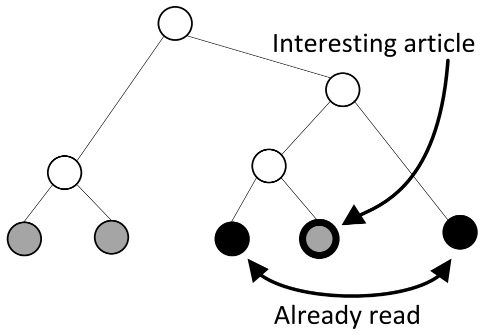

Fig. 7. Selecting interesting articles. Displayed articles are on the same branch as a candidate for recommendation.

In order to avoid overspecialization, we penalize the recommendation of overly similar interests, since otherwise the recommended articles would likely be very similar to those already viewed.

Because readers often have problems with long lists of recommendations [3], we choose articles that cover all relevant reader interests but are from distinct fields.

We also integrated time as an attribute in the computation of the list of recommendations, as it is a very important attribute, which indicates whether the interest that we discovered is outdated or current. Specifically in the news domain, time and recency are crucial. We store this additional information in our hierarchical structure and thus can find the latest article which was added for each branch of the tree. The time attribute is propagated as the maximum of two sub-branches. This way we efficiently identify the most recent article and the time when it was published. The interest is then as relevant as the last added article. We thus combine time relevancy and the content relevancy to create a combined list of recommended articles. Articles are not eliminated based on time recency, however aged articles are penalized. The tree contains all published articles (potentially interesting for some readers). We decide not to eliminate old articles because these could be still used for recommendation. We designed our method to work with huge amount of articles with low complexity. Even higher number of articles stored in the tree does not affect the time needed to recommend.

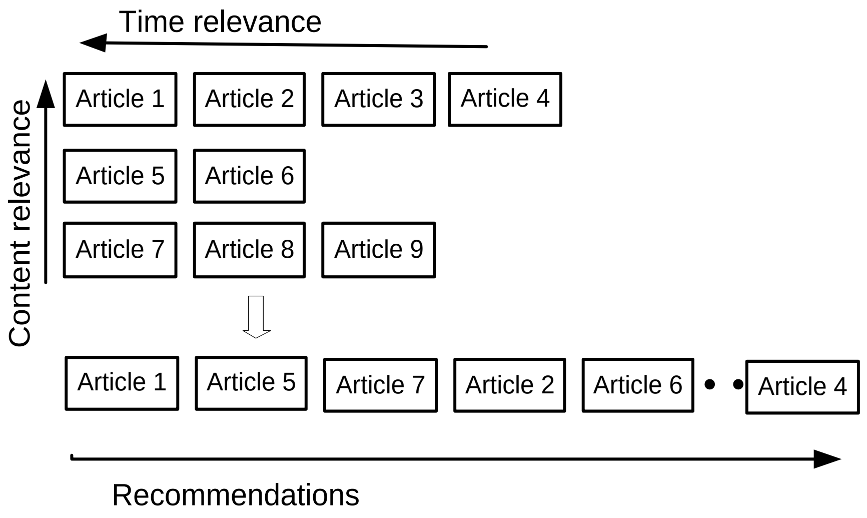

Fig. 8. Compiling the list of recommendations includes selection of most relevant articles based on content relevance and based on time. Articles in rows belong to the same field of interest; the most relevant interest is at the top. The most recent articles for each interest are on the left.

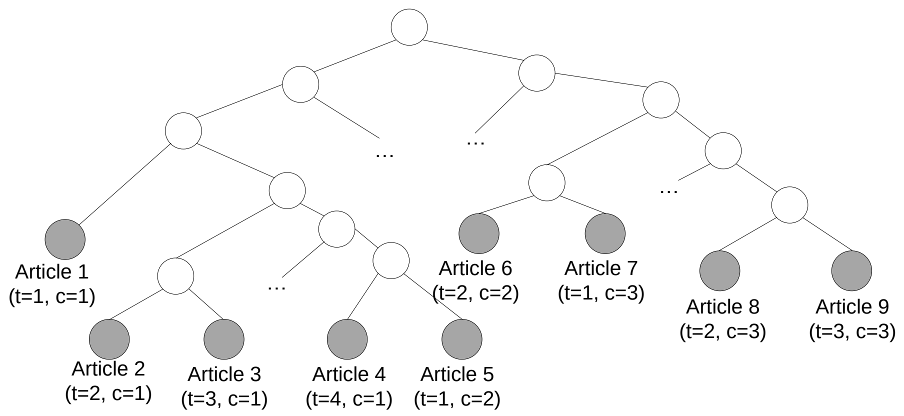

Fig. 9. Corresponding tree to the example of the recommendation compilation (Figure 8). Values *t* and *c* represent time and content relevancy for the specific user.

Figure 8 shows our method described in these steps:

- Selection of articles viewed by a reader
- Discovery of areas of interests in the tree
- Selection of unread articles for each interest
- Sorting articles by time for a particular interest

- Creation of an interest matrix
- Compiling the columns of the interest matrix into recommendations (Figure 8). Corresponding binary tree is shown in Figure 9.

The list of recommendations covers user interests but is designed to recommend at most 10 items per request to avoid choice overload [3] with articles covering several topics prioritizing recent articles.

We create the list of recommendations in real-time thanks to our similarity hierarchy which we use to represent relations between articles. In practice, we need to change the whole matrix only in cases where a user initiates a new session, or exhausts the recommendation list.

5. Evaluation

We proposed a method for effective content-based news recommendation. We designed this method to help readers to find interesting news and worked with more alternatives which we experimentally compared. We devised a data structure which we used to facilitate the sorting of text documents based on their similarity. Since the most interesting parameters for similarity computation are the used metrics, we experimented with different approaches to examine the effect on the quality of recommendation.

For our experiment, we used an interval of 14 days of user activity on the news portal SME.SK. We monitored articles which were viewed by users and have recorded over million unique users who had displayed over 12 million articles. We also added articles which were not viewed by users but were published during the interval. These additional articles belong to those which were not found by users but could be interesting. During preprocessing, we filtered out users who were not active and read only few articles during these 14 days, and also users who were too active (e.g., possibly the crawlers of search engines) thus reducing the noise caused by outlier users.

Our experiment was conducted using these records which were split into two subsets – one used for training our approach, and another for testing its accuracy. We used the training set to predict articles for every user who had been active in the test set. The predicted activity and real activity was then compared in via standard precision / recall metrics [12].

We started with simple bag of words which was used to represent and compare text documents. Later we worked with more sophisticated text preprocessing techniques, which have an important role in similarity search, because they can significantly reduce the search space. This part of the process is highly language dependent. Our experiments are performed in the Slovak language, which is one of the most complicated languages (declension of nouns, verbs etc.). The architecture of the system is variable, so preprocessing for Slovak language can be easily replaced by other languages and their methods (e.g., the Porter algorithm). Since maximum reduction of the article words dimensions and performance optimization are paramount, effective preprocessing played a critical role.

We first removed stop-words via a static list of words which are frequently used using TF-IDF under threshold output [28]. As the main part of the pre-processing of Slovak language articles we used text lemmatization, which cannot be algorithmically solved and needs to use a dictionary of lemmas. The result we received is a lemmatized (basic form) bag of words for every article.

Preprocessing of text is followed by the generation of the aforementioned vector representation for each article. In comparison to the whole article text which we firstly used to represent articles, we experimented with the vector components (metadata) and their weights (title words – 0.2, keywords – 0.3, category – 0.1, names and places – 0.4). We used different combinations of components to find the most appropriate components and how they affected the precision of our method during news recommendation. We placed these values as weight by experimentation. We used evolutionary algorithms as the approach for obtaining appropriate values. We placed fitness function as the similarity accuracy for articles.

We rebuilt the tree representation of article similarity and performed evaluation of our method for several combinations. We focused on the simple bag of words approach and then our vector representation where we observed the impact of names and places which were extracted from text on recommendation (see Table 4 for results).

We compared predicted user interest and articles viewed in the test set using a combination of category and section metadata, i.e. we did not compare exact articles but fields of interest which were covered by the combination of category and section where an article belonged. We chose this approach, because there were many articles on the same topic and users had no chance to read them all even if they were actually interested in the topic since one article on the same topic was often sufficient. To cover a topic we only compared the section and category of suggested articles and viewed articles. There were 420 valid options to choose this combination; therefore we guessed 1 combination out of 420 when predicting a topic.

Table 4. Comparison of the simple bag of words approach and vector based article representation during news recommendation.

Test interval [hours]		1 h	4 h	10 h	24 h	48 h
Bag of words similarity calculation	Precision	0.40	0.49	0.56	0.58	0.59
	Recall	0.71	0.60	0.44	0.32	0.25
	F1-Mesure	0.51	0.54	0.49	0.41	0.35
Vector based similarity calculation	Precision	0.35	0.42	0.52	0.54	0.66
	Recall	0.91	0.80	0.70	0.63	0.47
	F1-Measure	0.50	0.55	0.60	0.58	0.54

Our experiments indicate little to no improvement for shorter intervals. Predicting interests for 1 hour means to score only few articles. The average number of articles viewed by one user (from the set of active users) is 5 per hour. Precision in such cases is very low, hence recall is high. We recommend 10 articles what means that there is greater chance to cover all interests for small intervals.

We observed more interesting improvement for longer intervals. When recommending articles to cover topics in 48 consecutive hours the precision of our method is relatively high. Besides the reduction of article representation complexity, we also achieved improvement in precision and recall for these longer intervals. In Figure 10 we present distribution of precision (number of correct recommendations).

We implemented proposed approaches in Ruby 1.9.2 language on Rails 2 platform. Experiments were executed on Dual Core 2.1 GHz, 2GB RAM. Data structure was simulated in MySQL database using native indexing. The average adding time of article into the proposed tree representation was approximately 2s with logarithmical tendency of time increment. By this achievement we fulfilled our aim to keep the time complexity low, even if the number of articles is high. Time complexity is important especially in this domain of news recommendation where recency of articles which are recommended counts.

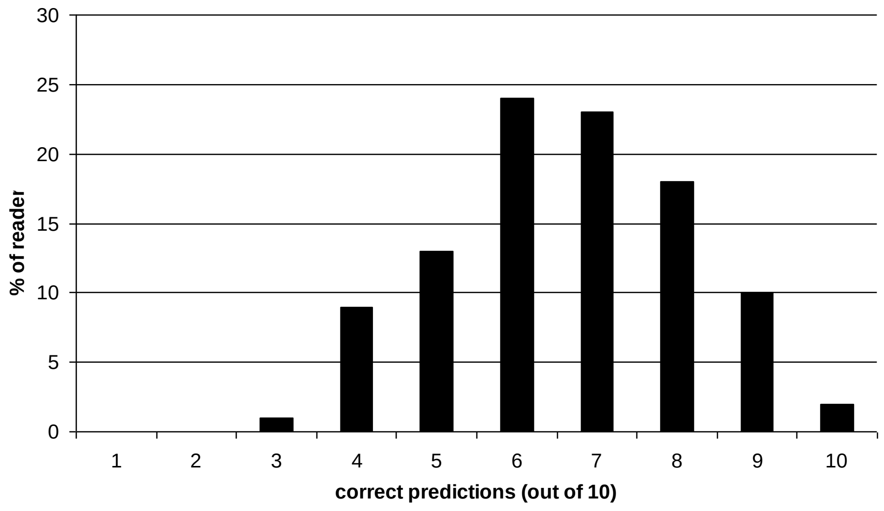

Fig. 10. Distribution of the readers by precision of the recommender. Shown values are for a 48 hour recommendation window over the test set.

6. Conclusion and Future Work

When recommending news several aspects have to be considered, such as the domain of news portals which is highly dynamic often requiring the use of content based methods. We compute article similarity at publication time instead of waiting for user ratings. Furthermore, the dynamic nature of the news domain is also visible in the number of articles (i.e., huge amounts of data) and the need for freshness (i.e., quick response to new articles). Recommender systems often use tags and meta-tags assigned to articles by authors or editors to provide an extra source of information, which can improve similarity computation. However, the so called "tunnel - vision" should be avoided and recommenders should consider various article topics in order to avoid overspecialization.

Our contribution is the effective and expressive article content representation, and binary tree representation of article similarity and user interests, which in combination enables real-time content-based recommendation in highly dynamic domains. The proposed approach ensures that overspecialization and very similar articles recommendation are avoided.

Every article vector consists of five sub-vectors based on the article part used for their construction. Every part has its own weight, which can be dynamically changed to rearrange similar articles list to enable fast personalization. Based on this computed similarity the binary-tree is constructed incrementally. Our tree representation allows us to mine user interests with various granularities depending on tree depth, and also overcome "tunnel-vision" and consider the freshness of articles in relevance computation.

We performed several experiments with real news portal data and have shown that the proposed method brings improvement especially for longer window intervals. There was no or only little improvement in the one and four hour test window, which can be explained by the lack of user activity for such a short time period. Beyond this, the proposed approach reduces the complexity of the similarity computation process significantly as we can compute recommendations in real time.

Limitations of proposed approach include well known cold start problem, since we cannot perform recommendation without sufficient user preference data. A combination with recommendation based on behavior of large number of readers (e.g. news popularity) is viable alternative to overcome this limitation. We currently work on mixed collaborative content-based recommendation as a merge of content-based approach described in this paper and collaborative approach which employs several strategies for recommending news based on implicit positive and negative feedback while reading [27]. A combination of several approaches including multiple-interests in the group recommendation approach is natural step in such research. We also plan to include in recommendation new sources of information about user context [16] such as time, mood, and location.

Acknowledgement. This work was supported by grants No. VG1/0675/11, APVV 0208-10 and it is a partial result of the Research and Development Operational Program for the projects Support of Center of Excellence for Smart Technologies, Systems and Services, ITMS 26240120005 and Research and Development. Operational Program for the projects Support of Center of Excellence for Smart Technologies, Systems and Services II, ITMS 26240120029, co-funded by ERDF.

References

1. Adomavicius, G., Tuzhilin, A. 2005. Toward the Next Generation of Recommender Systems. IEEE Trans. on Knowl. and Data Eng. vol. 17, no. 6, 734-749.
2. Billsus, D., Pazzani, M. 2000. User Modeling for Adaptive News Access. User Modeling and User-Adapted Interaction, vol. 10, nos. 2-3, (Feb. 2000),147-180.
3. Bollen, D., Knijnenburg, B. P., Graus, M. 2010. Understanding Choice Overload in Recommender Systems Categories and Subject Descriptors. In Proceedings of 4th ACM Conf. on Recommender Systems. Barcelona, Spain (Sept. 2010), 63-70.
4. Borges, H., Lorena, A. 2010. A Survey on Recommender Systems for News Data. Smart Information and Knowledge Management, 129–151.
5. Bouras, C., Tsogkas, V. 2009. Personalization Mechanism for Delivering News Articles on the User's Desktop. In Proc. of the 4th int. Conf. on Internet and Web Applications and Services (May 2009). ICIW. IEEE Computer Society, Washington, DC, 157-162.
6. Burke, R. 2002. Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction 12, 4 (Nov. 2002), 331-370.
7. Carvalho, C., Jorge, A. M., Soares, C. 2006. Personalization of E-newsletters Based on Web Log Analysis and Clustering. In Proc. of the IEEE/WIC/ACM Int. Conf. on Web Intelligence. IEEE Computer Society, WDC, 724-727.
8. Das, A., Datar, M., Garg, A., Rajaram, S. 2007. Google news personalization: scalable online collaborative filtering. In Proc. of the 16th Int. Conf. on World Wide Web, ACM Press, New York, 271–280.
9. Díaz, A., Gervás, P. 2005. Personalization in news delivery systems: Item summarization and multitier item selection using relevance feedback. Web Intelli. and Agent Sys. 3, 3 (Jul. 2005), 135-154.
10. Finkel, J.,R., Grenager, T., Manning, Ch. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In Proc. of the 43nd Annual Meeting of the Assoc. for Comp. Ling. (ACL), 363-370.
11. Gabrilovich, E., Markovitch, S. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In Proc. of the 20th int. Joint Conf. on Artificial Intelligence, Hyder., India, 1606-1611.
12. Ge, M., Delgado-battenfeld, C. 2010. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. In Proc. of 4th ACM Conf. on Rec. Systems. (Barcelona, Spain, Sept. 2010), 257-260.
13. Grcar, M., Mladenic, D., and Grobelnik, M. User profiling for the web. Computer Science and Information Systems 3, 2 (2006), 1-29.
14. Chen, Ch., Hong, Ch., Chen, S. 2009. Intelligent Location-Based Mobile News Service System with Automatic News Summarization. Int. Conf. on Environmental Science and Information Application Technology, 527-530.
15. Ingvaldsen, J. E., Gulla, J. A., Laegreid, T., Sandal, P. C. 2006. Financial News Mining: Monitoring Continuous Streams of Text. In Proc. of IEEE/WIC/ACM Int. Conf. on Web intelligence. Washington, DC, 321-324.
16. Jancsary, J., Neubarth, F., Trost, H. 2010. Towards Context-Aware Personalization and a Broad Perspective on the Semantics of News Articles. In Proc. of 4th ACM Conf. on Rec. Sys. Spain, 289-292.
17. Kroha, P.,Baeza-Yates, R., 2005. News classification based on term frequency. In Proceedings of the 16th Conf. on Database and Expert Sys. Apps, 428–432.
18. Kompan, M., Bieliková, M., 2010. Content-Based News Recommendation. In Proc. of the 11th Conf. EC-WEB, Springer, 61-72.
19. Lukashenko, R., Graudina, V., Grundspenkis, J. 2007. Computer-based plagiarism detection methods and tools: an overview. In Proc. of Int. Conf. on Computer

Systems and Technologies (Bulgaria, June 2007). CompSysTech '07, vol. 285. ACM, New York, NY, 1-6.
20. Melville, P., Mooney, R., Nagarajan, J. 2002. Content-boosted collaborative filtering for improved recommendations. In Proc. of 18th National Conf. on Artificial intelligence (Edmonton, Alberta, Canada). AAAI, Menlo Park, CA, 187-192.
21. Mooney, R. J. and Roy, L. 2000. Content-based book recommending using learning for text categorization. In Proc. of the 5th Conf. on Dig Libraries, TX, USA.
22. Pandya, A., Bhattacharyya, P. 2005. Text Similarity Measurement Using Concept Representation of Texts. LNCS 2776, 678-683.
23. Parapar, J., Barreiro, Á. 2009. Evaluation of Text Clustering Algorithms with N-Gram-Based Document Fingerprints. In Proc. of the 31st European Conf. on IR Research on Advances in information Retrieval (Toulouse, France, April 2009). LNCS, vol. 5478. Springer-Verlag, Berlin, Heidelberg, 645-653.
24. Ramos, J. 2000. Using TF-IDF to Determine Word Relevance in Document Queries. Tech. rep., Departament of Computer science. Rutgers University.
25. Sahoo, N., Callan, J., Krishnan, R., Duncan, G., Padman, R. 2006. Incremental hierarchical clustering of text documents. In Proc. of the 15th ACM int. Conf. on Information and knowledge management, NY, USA.
26. Su, X. and Khoshgoftaar, T. M. 2009. A survey of collaborative filtering techniques. Adv. in AI. 2009.
27. Suchal, J., Navrat, P. 2010. Full text search engine as scalable k-nearest neighbor recommendation system. In Proc. of the AI in Theory and Practice 2010. WCC. IFIP AICT 331, Springer, Boston, 165-173.
28. Tata, S., Patel, J.M. 2007. Estimating the Selectivity of TF-IDF based Cosine Similarity Predicates. SIGMOD Record, Vol. 36, 7-12.
29. Tintarev, N., Masthoff, J. 2006. Similarity for news recommender systems. In Proc. of the AH'06 Workshop on Rec. Systems and Intelligent User Interfaces.
30. Wang, X., Ju, S., and Wu, S. 2008. Challenges in Chinese Text Similarity Research. In Proc. of the 2008 Int. Symposiums on Information Processing (May 23 - 25, 2008). ISIP. IEEE Computer Society, Washington, DC, 297-302.
31. Wang, Q., You, S. 2006. Fast Similarity Search for High-Dimensional Dataset. In Proc. of the Eighth IEEE int. Symposium on Multimedia (Dec., 2006). ISM. IEEE Computer Society, Washington, DC, 799-804.
32. Wermter, S., Hung, C. 2002. Selforganizing classification on the Reuters news corpus. In Proc. of the 19th Int. Conf. on Computer Ling. - Vol 1 (Taipei), 1-7.
33. Wongchokprasitti, C., Brusilovsky, P. 2007. NewsMe: A case study for adaptive news systems with open user model. In: Proc. of Int. Conf. on Autonomic and Autonomus Systems, 2007. ICAS07, 69.
34. Wu, Y., Chen, Y., Chen, A. L. 2001. Enabling Personalized Recommendation on the Web Based on User Interests and Behaviors. In Proc. of the 11th Int. Workshop on Research Issues in Data Engineering. RIDE. IEEE Computer Society, Washington, DC, 17.
35. Yoneya, T., Mamitsuka, H., 2007. Pure: a pubmed article recommendation system based on content-based filtering. In Proc. of Int. Conf. on Genome Inf. 18, 267-276.
36. Ziegler, C., McNee, S. M., Konstan, J. A., Lausen, G. 2005. Improving recommendation lists through topic diversification. In Proc. of the 14th Int. Conf. on World Wide Web (Chiba, Japan, May, 2005). ACM, New York, NY, 22-32.

Maria,Bielikova received her Master degree (with summa cum laude) in 1989 and her PhD. degree in 1995, both from the Slovak University of Technology in Bratislava. Since 2005, she has been a full professor, presently at the Institute of Informatics and Software Engineering, Slovak University of Technology. Her research interests are in the areas of software web-based information systems, especially personalized context-aware web-based systems including user modeling and social networks. She co-authored over 60 papers in international scientific journals and she is editor of more than 30 proceedings, six of them published in Lecture Notes in Computer Science series of Springer. She is a member of IEEE Computer Society, ACM and International Society for Web Engineering.

Michal Kompan is currently a doctoral student at the Institute of Informatics and Software Engineering, Slovak University of Technology in Bratislava. He received master degree in 2010 from the same university. His research interests are in the areas of personalized recommendation systems for single or group of users, user modeling and social networks.

Dušan Zeleník is a doctoral student in Software Engineering study program at the Slovak University of Technology in Bratislava. He holds a master's degree in Software Engineering (2010) from the same university. His research interests are in areas of personalized systems, recommendation systems, user modeling and context. He is a member of the PeWe group which joins researchers and their interests at aforementioned university.

Received: April 4, 2011; Accepted: October 10, 2011.