Content-based News Recommendation

Michal Kompan and Mária Bieliková

Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies,Slovak University of Technology,
Ilkovičova 3, 842 16 Bratislava 4, Slovakia
kompan05@student.fiit.stuba.sk,bielik@fiit.stuba.sk

Abstract. The information overloading is one of the biggest problems nowadays. We can see it in various domains, including business, especially in the news. This is more significant in connection to web and news portals, where the quality of the news portal is commonly measured by amount of news added to the site. Then the most renowned news portals add hundreds of new articles daily. The classical solution usually used to solve the information overloading is a recommendation. In this paper we present an approach for fast content-based news recommendation, based on cosine-similarity search.

Key words: news, recommendation, vector representation, user model

1 Introduction

There are plenty of news portals on the web. Renowned and influential portal contains hundreds of new articles from the whole world added daily. These articles cannot be easily accessed. For example users of the biggest Slovak news portal SME.SK spend daily approximately 16 min 34 sec on the site, in usually two visits per day[1]. The amount of words on the websites have increased two times since year 2003 and we can see this effect applied to links, pictures, tables, advertisements etc. More than 60% respondents participating in IDC research said, that they face up the information overloading in more than half of the time (see Fig. 1).

One of the quality criteria for a good news portal is time spending by reading considering the amount of useful information acquisition. It is extremely important to enquire new information as quick as possible. The importance of fresh news can be easily seen on various non-news portals, where various shorten top news can be found.

We propose a method for content-based news recommendation, which uses our devised effective article representation. This representation is important when similar articles are computed. Fast similarity estimation plays the critical role in the high changing domains as news portals are. It is necessary to process a new article as fast as possible and start to this article recommendation, because of the high information value degradation. Finally we use these

[1] Source www.aimmonitor.sk - Association of Internet Media

similar articles to create recommended content based on the implicit user model. Our content-based method for recommendation is based on three steps - com-

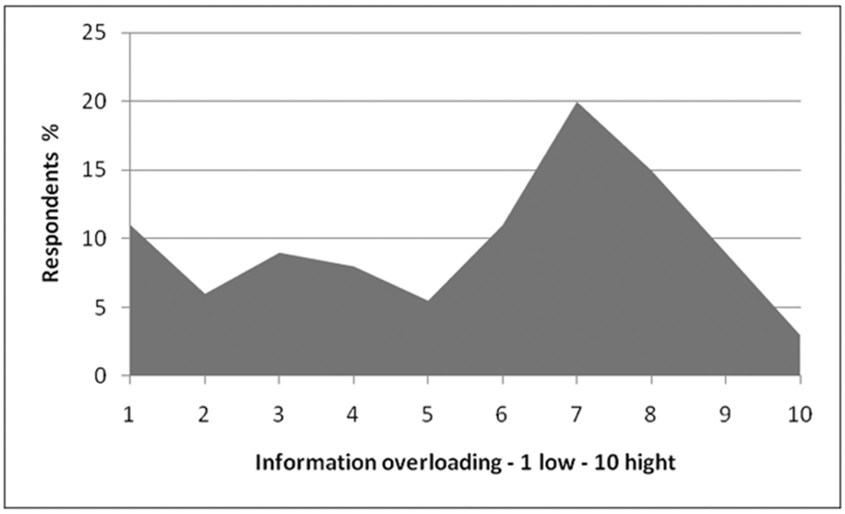

Fig. 1. Frequency of information overloading [IDC, autumn 2008, U.S., set of 500 respondents].

puting article similarity, creating a user model and the recommendation based on the first two steps (see Fig. 2). In the article similarity step it is necessary to pre-process every article to reduce word space. Then the article is represented in an effective vector representation, which is used in similarity computation. As a result of article similarity step we obtain a list of similar articles for every article in the dataset.

The user model is created based on implicit feedback extracted from server logs by identification of visited and recommended article for unique cookie.

Lastly is the recommended content from both similar articles and user model created. We will give deeply description for every step.

The paper is structured as follows. Section 2 describes state of work in the recommendation domain. In section 3 we provide overview of proposed vector structure representation. Section 4 describes our recommendation method. The evaluation of proposed method is described in section 5.

2 Related work

The recommendation is one of the actual research topics nowadays. Two basic approaches of the recommendation exists [10]. Traditional collaborative filtering

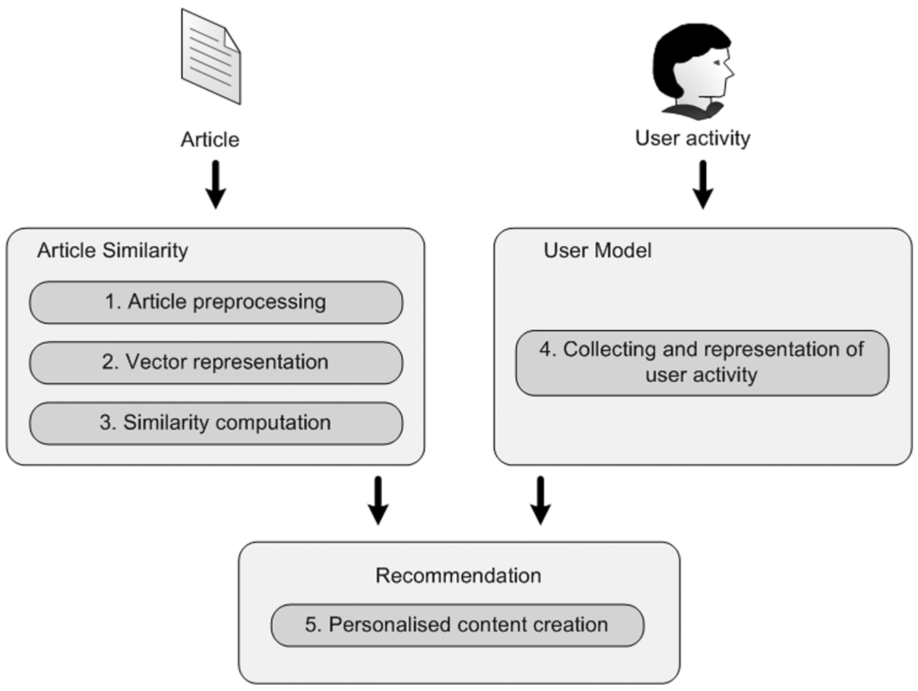

Fig. 2. Proposed news recommendation method.

accounts social element. Users are grouped into clusters based on their preferences, habits or content ranking. The problem of personalization is reduced to finding similar users and recommending new items to the users, which were visited and high ranked by other users in the same cluster.

The second approach of recommenders is based on the content-based filtering. The history of content-based filtering is connected with information retrieval and information research [1]. The main goal is to identify two similar items-create "clusters" of sites instead of users. It is necessary to map user profiles (user models) to specific site clusters. This type of filtering is successful in well structured domains like movies, news [11].

These two approaches are widely used and mixed together, which usually brings better results [9], [3]. For example, we can find similar sites and then estimate user rank prediction for sites, which were not visited. Also the combination of various approaches for every type is possible e.g. Google News [5].The main problem in the content-based filtering is effective and enough expressive representation of items (or articles). This is often done by means of text summarization [4], keywords extraction or by various categorization models [7]. These techniques are commonly used in English based systems and cannot be easily applied in other languages. Keywords extraction and summarization brings better results as other methods but are more time consuming. These methods cannot represent non-text documents without modification.

There are several recommender systems in the news domain. The problem within this domain which is rather similar to other business domains is extremely large amount of dynamically changed data. This causes that the recommendation is not provided directly over the whole data, when content-based recommendation is used [16], [17]. OTS system [16] use association rules to create "preference table" for every user. When there are a lot of new documents added daily, there is usual to not compute recommendation lists real-time [17]. TRecom is a promising method for content-based recommendation, when uses binary-tree representation of similarity and user's preferences [18]. Brusilovsky [15] has shown that explicit filled and open user model in the news domain brings usually worse results. Some systems have involved user location into recommendation systems, where recommendation list is created depending on user location [6].

3 Article similarity computation

For fast similarity estimation we propose effective vector article representation. This representation consists of six basic parts:

- *Title.*Lemmatized words from article title (approx. 5 words - 150 000 Slovak article corpus.) This should be good describing attribute in the most occurrences.
- *TF of Title words in the article content.*We use term frequency to compute article relevance. If the article name is abstract and do not correspond to article content, we can easily reveal this situation (threshold). Term frequency is computes as follows:

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

  where tf_i is term frequency for term i (term from article title) and n_i is number of occurrences of term i in the document (article content) and $\sum_k n_k$ is the sum of numbers of all terms in document.
- *Names and Places.*We extract names and places from article content. There exists several names or places extractors for English language. We use simple approach to detect these items. As name or place is marked word starting with an upper letter and there is no sentence end before (dot, question mark etc.). This method can easily extract most of names and places occurred in the article (precision = 0.934, recall = 0.863).
- *Keywords.*We store 10 more relevant keywords. Several news portals define list of keywords for every article. These keywords are unfortunately on various abstract levels for various news portals. We introduced our own keywords list based in TF-IDF computation (150 000 Slovak news articles from news portal SME.SK). We adopt a part of speech tagging and removed any words except nouns and names[2].

[2] JULS dictionary - Slovak Academy of Sciences

– *Category.*Consists of "tree-based" category vector with weights. This vector is constructed based on specific news portal structure hierarchy (optional). This is useful, when not enough similar articles are found. The weight for every category is computed as:

```
n=1
For i=|Category| downto 0 do
 weight[i]=1/n
 n=n*2
end
```

– *CLI.*Coleman-Liau Index provides information of understandability of the text. This vector part is not important for standard similarity computation, but it is important in the results rearrangement. Our hypothesis is that the user wants to read articles of one similar level of understanding. This method is able to distinguish between two articles with similar title and different content ("Jaguar" - animal vs. car). CLI can be easily computed based on this formula [8]:

$$CLI = 5.89 \times \left(\frac{characters}{words}\right) - 29.5 \times \left(\frac{sentences}{words}\right) - 15.8 \tag{2}$$

When using this article representation, we can store an article in the vector no longer than 30 items in most of occurrences. Example of proposed representation is given in Table 1. For the purpose of similarity computation, we use cosine

Table 1. The example of vector article representation.

Vector part	**Weights**
Title	transplantácia_0.5 tvár_0.5
TF of title words in the content	transplantácia_0.0178571428571429 tvár_0.0714285714285714
Category	Sme.sk_0.5 PRESS_FOTO_1.0
Keywords	klinika_0.0357142857142857 povrch_0.0178571428571429 nos_0.0178571428571429 zub_0.0178571428571429 nerv_0.0178571428571429 svalstvo_0.0178571428571429 pacientka_0.0178571428571429 rozsah_0.0178571428571429
Names/Places	Cleveland_1
CLI	0.2543

similarity [14], which is widely used in the information retrieval tasks. Our vector consists of 6 sub-vectors with weights so there is need to extend standard cosine similarity as:

$$similarity = \frac{\sum_{j=1}^{m} \sum_{i=1}^{n} a_{ji} b_{ji}}{\sqrt{\sum_{j=1}^{m} \sum_{i=1}^{n} a_{ji}^2} \sqrt{\sum_{j=1}^{m} \sum_{i=1}^{n} b_{ji}^2}} \tag{3}$$

where m is number of vector parts (6 in our method) and n is number of vector items.The similarity definition in recommendation systems is a difficult task. We can define similarity based on news content (like plagiarism task), or based on "topic" or "affair". This is extremely important when a recommendation list is created. Our method respects each of these types. We can easily redefine our similarity with simple changing the weights for vectors parts and adjust it for various recommender methods.

3.1 News Pre-processing

Text pre-processing holds an important role in the process of similarity search, because it can significantly reduce word space. This part of the process is high language depending. Our experiments are provided 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 pre-processing for Slovak language can be easily replaced by other languages and their methods (e.g. Porter algorithm[3]). For the speed of next computations, plays pre-processing a critical role. There is need to maximum reduction of article words dimensions.

The first task is to remove stop-words. We used static list, which can be replaced by TF-IDF output [12]. This method can identify commonly repeated words over the dataset.

As the main part of the pre-processing of Slovak language articles we used lemmatizing of the text. There is problem with algorithmic solution for this process, which can be solved by using dictionary of lemmas. For the purpose of lemmatization we use dictionary of lemmas (600 000 records). The result we receive is lemmatized (basic form) bag of words for every article.

It is necessary to note, that we removed any punctuation except sentences ends. We use dots as a fast name or place indicator - when we check if there is a dot before an uppercase letter, and if not, it is probably personal name, or place etc. Names and place extractors are standard tasks in information retrieval. As we mentioned above, used approach brought sufficient results and can be simply substituted by more sophisticated methods.

After keywords extraction we do not need whole article content anymore. We can safely delete all words except Title words obtained in the article content. Then for every processed article we have this list of words:

– Lemmatized article Title

[3] The Porter Stemming Algorithm page maintained by Martin Porter. www.tartarus.org/ martin/PorterStemmer

– Lemmatized words from Content (which were included in the Title)
– 10 most relevant Keywords
– List of Names and Places

Pre-processing methods we described above can significantly reduce number of words stored for every article up to 80%.

4 Recommendation

The most important part of proposed method is recommendation step (Fig. 3). For recommendation creation we need two lists as an input. First is list of 10 most similar articles for every article computed as we described above. Second list is list of visited articles for every user based on cookie. In this list we need to distinguish between articles visited but not recommended to users and articles visited and recommended before which can be easily done by extending article URL with special attribute. Firstly we have to define number of articles to

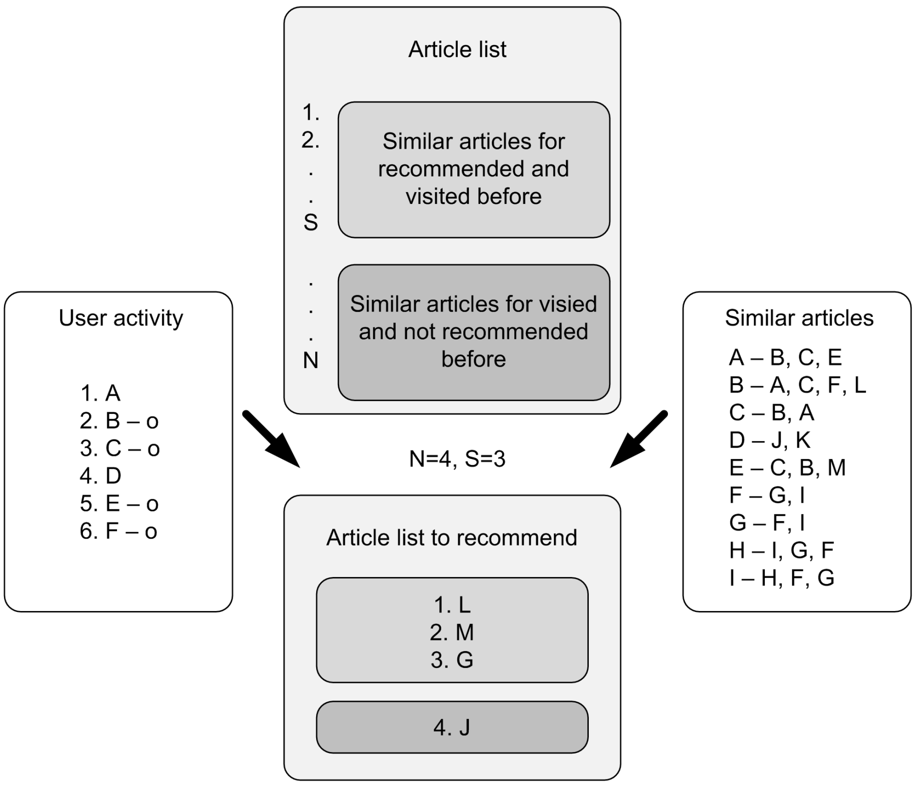

Fig. 3. Recommendation method steps.

recommend (length of list to recommend). As we can see in Fig. 3 list of articles to recommend consists of two sub-lists:

– List of similar articles for visited and not recommended (S)
– Similar articles for visited and before recommended (N-S)

The ratio of this list is dynamically computed as:

$$S = N\left(1 - \frac{Nr}{V}\right) \tag{4}$$

where S is number of similar articles for visited and not recommended articles, N is number of articles to recommend. Nr represents number of visited not recommended articles during the last session and V is number of visited articles together. For the proposed method, two sessions are distinguished as 1 hour break between visits.

Recommendation list is then computed for every part separately as follows:

```
foreach cookie do
 visited = get visited articles list
 visitedRec = get visited and recom. articles list
 foreach visited do
  if randomNum > random treshold
   listPart1 = get first non visited article from computed
               similarity list
  else
   listPart1 = get random non visited article
  end
 end
 foreach visitedRec do
  listPart2 = get first non visited article from computed
              similarity list
 end
 listToRecommend = listPart1[1..N] + +listPart2[1..M]
end
```

When there is not enough user activity (does not mean "cold start") we use random article assignment. In this manner we can easily react to user's most recent preferences. For the list of recommended and visited articles we also introduced a coincidence - where user obtains a random article to the recommendation list.

Fig. 3 presents an example of recommended content creation. We have a list of user activities where "-o" attribute indicates whether the article was or was not recommended. Based on this we obtain list of visited articles and list of visited and before recommended articles. Then when we want to recommend 4 articles (*N=4*), we will obtain ratio 3:1 for sub-lists (similar articles for visited and recommended, similar articles for visited and not recommended before).

As we can see in our example, we have 4 visited and recommended articles *B, C, E, F*. We have found non visited article from the list of similar articles for every of these 4 articles. There is only one non visited similar article L for article B. This is repeated until the "before recommended" list is full. In the case when there do not exist non visited article in the similar list (article C), we skip this article, because the user saw all relevant articles for this "topic" already.

In our example there are 2 non recommended but visited articles, but there are no non-visited articles for A - method will skip this article and will recommend first non visited for article D. In this manner we obtain a full list of 4 articles to recommend.

Dynamical computation of the ratio between sublist allows us to adapt for actual user activity and preferences. If the user is not interested in recommended articles and he uses other portal navigation, the size of the first sub-list is decreasing while second part will increase respectively.

Our method stores "article age" for every recommended article. This number represents how long have been article recommended. If user does not visit this article for a defined time (number of recommendations) is this article deleted from the recommendation list as not interesting.

User activity list consists of pairs cookie - visited article. We use implicit user model representation, where there is no need to involve users into various forms completing or need of logging etc.

5 Experimental results

Proposed method was implemented within news recommendation system in the research project SME-FIIT [2].We evaluated the similarity computation over 10 000 articles from the Slovak news portal SME.SK, which is equivalent to one week time period. For this window we are able to estimate the similarity in 2-3 seconds (2,6 MHz Pentium, 4Gb RAM, Ruby). The pre-processing takes approximately 20 seconds for the whole dataset. Then for the new article, when pre-processing is necessary, the whole computation process takes approximately 22s. When we need only re-estimate similarity with changed vectors parts weights is this process really fast as we mentioned above.

The accuracy of the similarity computation method was computed based on two datasets. The first one consists of 1 000 articles from news portal SME.SK. Every article from the dataset has assigned at least one similar article. These similar articles were obtained from the news portal, where there are mostly one or two similar articles quoted in the article footer. These similar articles are obviously chosen by the article author, which does not mean that there are not more similar articles.

The second dataset was the manually annotated dataset, which consists of 100 articles in 5 levels of similarity, so we obtained 10 000 article pairs with similarity level. Our method computed the list of similar articles for every article in the dataset. We compared these datasets to our method - the list of similar articles computed by our method and the list of similar obtained from one of two datasets with respect to order (more similar articles first). We calculated precision and recall and F-Score for every dataset and the method. Results were compared to standard text mining method TF-IDF (whole article content) as shown on Table 2.

Table 2. Similarity computation evaluation.

Dataset	**SME.SK**		**Manually annotated**	
Method	Our method	TFIDF	Our method	TFIDF
Precision	0.165	0.091	0.700	0.511
Recall	0.202	0.117	0.816	0.587
F-score	0.182	0.102	0.753	0.546

The dataset SME.SK is created based on ”similar article” data (none, one or two) in the articles footers. These similarities are assigned by article’s authors intuitively and often this choice does not mean not the only possibility but also one of the best matching articles. This is reflected in the results as we obtained only 0.182 F-score. Providing manual check we found out that our method in most cases founded more similar (and relevant) articles as the authors assigned. This indicates that manual similarity articles list creation by the article authors can be improved by our method.

We also computed standard deviation based on similarity levels. We mapped cosine similarity range $< 0, 1 >$ to five similarity levels used in our manually annotated dataset. The worst standard deviation we obtained - 1.21 ”similarity level” is an acceptable rate in the field of news recommendation.

6 Conclusion

In this paper we provided the overview of short and high representative article vectors, which can be used for similarity search and real content-based recommendation in a large and dynamically changing datasets and domains. A key future of this method is short article representing vector. In pursuance of these vectors can be computed similarity between articles (text or non-text content) in a fast way. Every article vector consists of 6 sub-vectors based on 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. This approach is easily extendible to other languages.

Weights were found by using evolution algorithms for every vector part, to obtain the best result. As an example, use of proposed representation brings 4 times better precision than use only article title, and at least 1.4 time better results as use only keywords. By considering the category part we improve precision only 1.15 time, but on the other hand it can be useful when ”no similar” article is in the dataset.

Once the similarity is computed, further recommendation is created. User preferences are collected implicitly via server's logs. A recommended list consists of two sub lists, where the first one represents similar articles to the visited and already recommended. The second sublist is based on similar articles for visited but not recommended before. In this way we can easily adapt to user preferences. The ratio between these two sub-lists is dynamically computed.

Proposed vector representation is a promising method for the fast news similarity computation to allow a real time recommendation. We plan to make improvements on the precision and the recall, for example by using more sophisticated keywords extraction methods etc. and evaluate whole recommendation method by it's implementation to existing news portal. Furthermore we expect significantly better results when combining our approach with TRecom [18] method or collaborative recommendation [13].

Acknowledgements

This work was partially supported by the grants VEGA 1/0508/09, KEGA 028-025STU-4/2010 and it is the partial result of the Research & Development Operational Programme for the project Support of Center of Excellence for Smart Technologies, Systems and Services, ITMS 26240120029, co-funded by the ERDF.

References

1. Adomavicius, G., Tuzhilin, A., 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, pp. 734-749.
2. Barla, M., Kompan, M., Suchal, J., Vojtek, P., Zeleník, D., Bieliková, M. News recommendation. In Proc. of the 9th Znalosti, pp. 171-174. 2010.
3. Bouras, C., Tsogkas, V., 2009. Personalization Mechanism for Delivering News Articles on the User's Desktop. In Proc. of the 2009 Fourth int. Conf. on internet and Web Applications and Services (May 24 - 28, 2009). ICIW. IEEE Computer Society, Washington, DC, pp. 157-162.
4. Dakka, W., Gravano, L. 2007. Efficient summarization-aware search for online news articles. In Proc. of the 7th ACM/IEEE-CS Joint Conf. on Digital Libraries. JCDL '07. ACM, New York, NY, pp. 63-72.
5. Das, A. S., 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. WWW '07. ACM, New York, NY,pp. 271-280.
6. Chen, Ch., Hong, Ch., Chen, S., 2009. Intelligent Location-Based Mobile News Service System with Automatic News Summarization. International Conference on Environmental Science and Information Application Technology, pp. 527-530.
7. Kou, H., Gardarin, G., 2009. Keywords Extraction, Document Similarity and Categorization. Tech.rep. No.2002/22, PRiSM Laboratory of Versailles Univ.
8. McCallum, D. R, Peterson, J. L., 1982. Computer-based readability indexes. In Proc. of the ACM '82 Conf. ACM 82. ACM, New York, NY, pp. 44-48.

9. Melville, P., Mooney, R. J., Nagarajan, 2002. Content-boosted collaborative filtering for improved recommendations. In Proc. of 18th National Conf. on Artificial intelligence (Edmonton, Alberta, Canada). AAAI, Menlo Park, CA, pp. 187-192.
10. Mobasher, B., Anand, S., S., 2005 Intelligent Techniques for Web Personalization: IJCAI 2003 Workshop, ITWP 2003. (Lecture Notes in Artificial Intelligence). Springer-Verlag New York, Inc.
11. Pazzani, M., Billsus, D., 2007. Content-based recommendation systems, pp. 325-341.
12. Ramos, J., 2000. Using TF-IDF to Determine Word Relevance in Document Queries. Tech. rep., Departament of Computer science. Rutgers University.
13. Suchal, J., Návrat, P., 2010. Full text search engine as scalable k-nearest neighbor recommendation system. In Proc. of the Artificial Intelligence in Theory and Practice 2010. World Computer Congress. Springer Boston.
14. Tata, S., Patel, J.M., 2007. Estimating the Selectivity of tf-idf based Cosine Similarity Predicates. SIGMOD Record, Vol. 36, pp. 7-12.
15. Wongchokprasitti, C., Brusilovsky, P., 2007. Newsme: A case study for adaptive news systems with open user model. In: Autonomic and Autonomous Systems, 2007. ICAS07. Third Int. Conference, pp. 69.
16. 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.l Workshop on Research Issues in Data Engineering. RIDE. IEEE Computer Society, Washington, DC, 17.
17. Yoneya, T., Mamitsuka, H., 2007. Pure: a pubmed article recommendation system based on content-based filtering. Genome informatics. International Conference on Genome Informatics 18, pp. 267-276.
18. Zeleník, D., Bieliková, M., 2009. Dynamics in hierarchical classification of news. In Proc. of the 4th Work. on Intel. and Knowledge oriented Technologies (WIKT 2009), pp. 83-87.