User Profiling in Text-Based Recommender Systems Based on Distributed Word Representations

Anton Alekseev[1] and Sergey Nikolenko[1,2,3(✉)]

[1] Steklov Mathematical Institute at St. Petersburg, St. Petersburg, Russia
sergey@logic.pdmi.ras.ru
[2] National Research University Higher School of Economics, St. Petersburg, Russia
[3] Deloitte Analytics Institute, Moscow, Russia

Abstract. We introduce a novel approach to constructing user profiles for recommender systems based on full-text items such as posts in a social network and implicit ratings (in the form of likes) that users give them. The profiles measure a user's interest in various topics mined from the full texts of the items. As a result, we get a user profile that can be used for cold start recommendations for items, targeted advertisement, and other purposes. Our experiments show that the method performs on a level comparable with classical collaborative filtering algorithms while at the same time being a cold start approach, i.e., it does not use the likes of an item being recommended.

Keywords: User profiling · Recommender systems · Distributed word representations

1 Introduction

In the modern Web, with the advance of social interactions between users and full-scale data mining of all information related to users, user profiling has become a very important problem. In this context, user profiling means converting the recorded user behaviour into a certain set of labels or probability distributions that capture the most important aspects of the user that can be further used for making new recommendations, providing targeted advertisement, and various other purposes. User profiles can incorporate real-life knowledge such as demographic information (age, gender, location etc.) or attempt to infer it from user behaviour. However, the holy grail of user profiling is to concisely represent a user's topical interests, preferably as narrowly as possible, with obvious applications to targeted advertising and new recommendations. The methods of summarizing information about users lie at the core of many personalized search and advertisement engines and various recommender systems. Being able to make predictions based on appropriately summarized prior user-system interaction allows, among other things, to alleviate the so-called *cold start* problem,

D.I. Ignatov et al. (Eds.): AIST 2016, CCIS 661, pp. 196–207, 2017.
DOI: 10.1007/978-3-319-52920-2_19

which is one of the main problems of recommender systems: how do you recommend a new item that has not been rated before or has had very few ratings? Given user profiles and a way to match the new item to these profiles, one can make recommendations when collaborative filtering is inapplicable.

This motivation ties in well with full-text recommendations. When users interact with items that have actual text associated with them, this allows for a possibility to infer topical user profiles based on automated mining of the texts they interact with. This problem has become especially relevant in recent years due to the growth of the social Web, where users interact with various texts all the time, not only reading but actively rating them. And as for possible solutions, recent advances in natural language understanding, especially in distributional semantics, provide many promising new methods for this problem. This is precisely the path that we take in this work, using topical clusters based on distributed word representations to construct user profiles.

The paper is organized as follows. In Sect. 2, we describe the problem setting in detail and survey related work. In Sect. 3, we describe in detail the novel approach we present in this work. Section 4 shows experimental results that validate our approach, and Sect. 5 concludes the paper.

2 Related Work

2.1 User Profiling in Recommender Systems

User profiling is a special case of user modeling. For general reviews of the field and key papers, we refer to [1–5]. Specific techniques that have been applied to represent user interests in content-based and hybrid recommender systems include, for example, relevance feedback and Rocchios algorithm [6,7], where a user profile is represented as a set of words and their weights, penalized if the retrieved textual item is uninteresting, as in [8]. Ontologies and encyclopaedic knowledge sources have been used, e.g., in Quickstep and Foxtrot systems [9] that recommend papers based on browsing history, automatically classify the topics of a paper and make use of relations between the topics in ontology to obtain their similarity; rank is computed based on the correlation of user profile topics and estimated paper topics. Nearest neighbours are often used in such systems; e.g., DailyLearner [10] stores tf-idf representations of recently liked stories in a short-term memory component, using it to recommend new stories [6,7]. Decision rules have been used, e.g., in the RIPPER system [11,12], where rules are a conjunction of several tests against items features. Interpretable predicted user characteristics are also often utilized in practice; cf., e.g., Yandex.Crypta.

2.2 Distributed Word Representations

Distributed word representations have become important for natural language processing with the rise of NLP systems based on neural networks; models for individual words, known as word embeddings or distributed word representations, map the words into a low-dimensional semantic space, trying to map

semantic relations to geometric ones in that space. To train word representations, a model with one hidden layer attempts to predict the next word based on a window of several preceding words or words that surround it (skip-gram); this approach has been applied, for instance, in the Polyglot system [13], GloVe [14] Recent studies on the performance of various vector space models for word semantic similarity evaluation [15] demonstrated that compositions of models such as GloVe and Word2Vec as well as unsupervised one-model approaches show reasonable results for the Russian language.

3 User Profiling with Distributed Word Representations

3.1 Problem Statement and General Outline

In this work, we propose a novel method for user profiling in full-text recommender systems, constructing a user profile as an interpretable summary of the user's interests that can also be utilized for recommending new items solely based on the prior state of the system. We begin with a brief outline of our approach.

First, we cluster all word representations trained on an external corpus (see Sect. 3.2). We have obtained high-quality clusters that are easy to interpret, so they were chosen to serve as a basis for user profiling; a user would be characterized by his or her affinity to these clusters.

For the recommender system, we used a large dataset from the "Odnoklassniki" online social network; we used group posts (texts in online communities written by their members) and individual user posts (texts published by a user on his/her profile page) as full-text items and user likes for these posts as ratings. There are two important obstacles along this way.

1. First, the dataset contains only positive signals from the users (likes), which is common in real life recommender systems but which makes it hard to train. While recommender systems based on such implicit information do exist, e.g., recommender systems based on max-margin nonnegative matrix factorization [16], it is unclear how to adapt them to full-text recommendation and user profiles in the semantic space.
2. Whatever technique one tries for the problem, user profiles always tend to be dominated by clusters/topics consisting of common words that occur often in the texts of various topics, but are useless for recommendations.

The second problem was especially hard to solve; we solved it with a novel approach to user profiling based on logistic regression trained multiple times on random subsets of the dataset; this approach is described in Sect. 3.3.

3.2 Clustering Word Vectors

In our experiments, we use a skip-gram *word2vec* model of dimension 500 trained on a large Russian language corpus [15,17]; the corpus consisted of:

– Russian Wikipedia: 1.15 M documents, 238 M tokens;
– automated web crawl data: 890 K documents, 568 M tokens;
– texts from the *lib.rus.ec* library: 234 K documents, 12.9 G tokens.

A large collection of general-purpose texts ensured good resulting distributed representations; for an in-depth description of the model we refer to [15,17].

To get a finite set of possible user interests or document topics, we clustered the word vectors directly. Note that while for some other applications topic modeling [18,19] might prove to be more useful, but in our case the basic underlying texts were too short and of too poor quality to hope for a good topic model, decisions regarding the topics would often have to be made on the basis of one or two keywords. Besides, we wanted to develop a top-down general approach that would be applicable directly even without a large and all-encompassing collection of texts available directly in the recommender system.

The embeddings of terms that occurred in our social network posts dataset resulted in 111281 vectors to be clustered in the $\mathbb{R}^{500}$ space. We have tried several methods for large-scale data clusterization, including Birch [20], DBSCAN [21], and mean shift clustering [22], but despite being generally able to process 100K+ items, these methods have proven to be not fast enough for high-dimensional data (for dimension 500 in our case), coupled with a large number of clusters (several thousand). The best option was still provided by classical k-means clustering. We applied mini-batch k-means that samples subsets of data (mini-batches) and then applies standard k-means to then: they are assigned to centroids, and centroids are "moved" to actual centers; the updates are done stochastically, after every mini-batch [23]. For initialization, we used the k-means++ approach that initializes cluster centers as far from each other as possible and then applies standard k-means to a random data subset to refine initialization [24].

Table 1 shows sample clusters together with their *idf* (inverse document frequency) values. It is clear that the most frequent clusters largely consist of common words that do not represent any specific topic that could be used for recommendations; they will be our major problem in the next section.

Table 1. Sample clusters.

IDF	Terms
3.276	Decide family leave buy parent read case week ...
4.469	Smile work answer appreciate state goal given inside brain remind ...
5.703	Comment Quran culture union Kim German note interview East forum historical ...
5.902	Salt pepper garlic sour-cream greens vegetables carrot cucumber ...
6.126	Pain disease shock depression abortion cardiac dense muscular insomnia ...
7.608	Stick axe thunder arrow sword boomerang shield spike steel armor paddle ...
8.239	Lead fly move once drive run walk ...
9.650	Bacteria molecule spermatozoid leukocyte chromosome mitochondria amoeba ...
11.004	Scaffold gallows pardon torture quartering hanging beheading ...

3.3 Estimating Cluster Affinity with Resampling Logistic Regression

We begin with the following notation:

- D is the set of documents in concern,
- C is the set of clusters,
- T is the set of all words in concern,
- T_c is the set of words in a cluster $c \in C$,
- word2vec : $T \to \mathbb{R}^d$ is the function assigning each word its embedding,
- $\mathrm{df}(t)$ is the number of documents the word $t \in T$ occurs in,
- clust : $T \to C$ is a function returning the cluster of a word,
- I_u^{like} is the set of all items user u liked.

To produce user profiles, we first constructed fixed-dimensional vector representations of documents $v_d \in \mathbb{R}^d$ for each document $d \in D$, representations of clusters of documents $v_c \in \mathbb{R}^d$ for each cluster $c \in C$ based on the representations of documents, and finally representations of users $v_u \in \mathbb{R}^d$ for each user $u \in U$ based on representations of the documents they liked and the corresponding clusters; in our experiments, $d = 500$. To build vector representations, we used a straightforward approach based on averaging and tf-idf weighting. Suppose that we know word2vec word embeddings for a large proportion of words in our data (not all due to typos, proper names and the like), $v_c = \frac{1}{|T_c|}\sum_{t \in T_c} \text{word2vec}(t)$ for each $c \in C$. Then we define

$$\mathrm{df}_c = \sum_{t \in T_c} \mathrm{df}(t), \quad IDF_c = \log\left(\frac{\sum_{c^* \in C} \mathrm{df}_{c^*}}{\mathrm{df}_c}\right), \quad v_d = \sum_{t \in d} \frac{IDF_{\text{clust}(t)} \cdot v_{\text{clust}(t)}}{IDF^d_{sum}},$$

and $IDF^d_{sum} = \sum_{t\ ind} IDF_{\text{clust}(t)}$. Finally, the user representation is

$$v_u = \sum_{d \in I^u_{liked}} \frac{\sum_{t \in d} IDF_{\text{clust}(t)} \cdot v_{\text{clust}(t)}}{Z},$$

where Z is a corresponding normalization value.

Then we constructed a new representation of a document, designed as a vector of cluster likelihoods $p(c|d)$; namely, for every document $d \in D$ and every cluster $c \in C$ we computed

$$L(d|c) = e^{\frac{-\|v_d - v_c\|}{2\sigma^2}}, \quad p(c_i|d_j) = \frac{L(d_j|c_i)}{\sum_d L(d|c_i)}.$$

Then, to construct the profile of a user u by his or her set of liked items I_u^{like}, we repeated the following procedure N times independently (in the experiments below, we used $N = 100$):

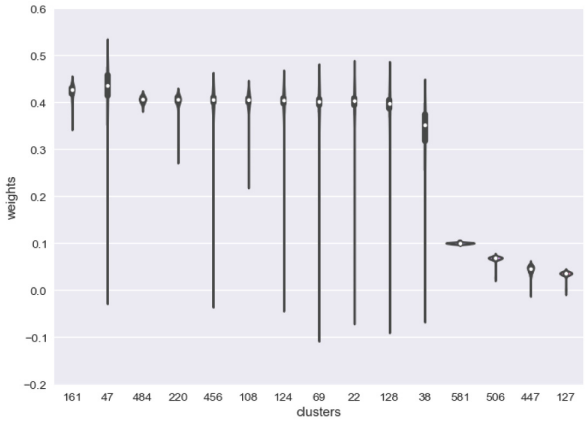
(i) on step k, draw a random sample from the documents the user u didn't like, taking the size of the sample equal to the number of documents the user actually liked; we denote it by $I^{\text{dislike}}_{u,k}$;

(ii) train logistic regression with the following data: I_u^{like} are the positive examples, $I_{u,k}^{\text{dislike}}$ are the negative examples, and the features are document affinities to clusters $p(c|d)$, $d \in I_u^{\text{like}} \cup I_{u,k}^{\text{dislike}}$;
(iii) as a result of this logistic regression, we get a set of weights $w_{u,c,k}$ for each cluster c.

Then, for every user u his or her profile is defined as the parameters of the normal distribution for every weight $\{(c, \mu_{u,c}, \sigma_{u,c})|c \in C\}$, each $(\mu_{u,c}, \sigma_{u,c})$ trained on the set $\{w_{u,c,k}|c \in C\}$.

In other words, logistic regression here is used to approximate the probability of a like; it trains a hyperplane separating liked items from items that have not been liked in the semantic feature space. This simple approach would be sure to fail if we simply trained liked documents against non-liked documents since the dataset is vastly imbalanced (a single user can be expected to view but a tiny fraction of all items); hence the random sampling of non-liked documents.

However, there is one more purpose to the random sampling apart from balancing the problem. We would like to solve the problem of common-word clusters, clusters that contain common words, are ubiquitous in the dataset, and tend to dominate all user profiles simply because by random chance, a user will like more than their fair share of some of common word clusters. In randomly



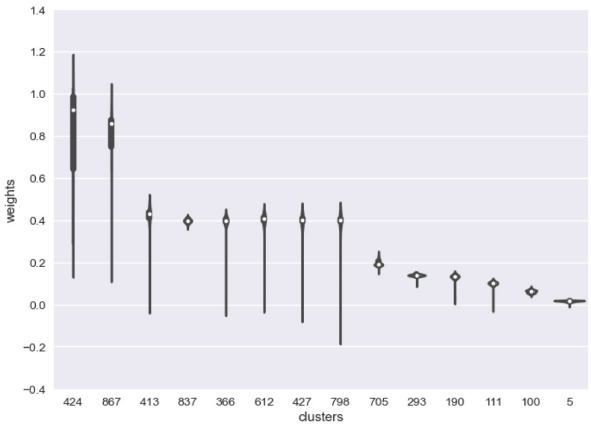
#	μ	σ	Words
161	0.422	0.017	uni din tel tine adam riga van eden loc publ etc judo art professor polis...
47	0.421	0.074	Nido Josep Jordi Victor Fbio Paulinho Avito Oswaldo Oliveira Julio...
484	0.407	0.007	Virgo Aries Taurus Dey hill branch statue strand Hon patron figure Chakhon...
220	0.405	0.014	Christ soul spirit God pray purify verse forgiven salvation devote Saturday...
108	0.396	0.035	Daniel Vladimir chronicle Tver Svyatoslav Ryazan Novgorod Pskov veche...
124	0.391	0.062	ask talk call listen report try calm assuring prompt pleading convince...
69	0.386	0.081	minute managed approach this autumn song male children ages anxiety...

Fig. 1. Sample user profile produced by resampling logistic regression.

sampling the negative examples, we get a certain distribution of "concentrations" of different clusters in the negative example. Note that:

– "topical" clusters that contain rare words will seldom occur in negative examples and thus the variance of the resulting weight distribution $\sigma_{u,c}$ with respect to these clusters will be low;
– "common word" clusters that contain words that are widely distributed across the entire dataset will sometimes appear more often and sometimes less often in the negative examples, and thus the variance of the resulting weight distribution $\sigma_{u,c}$ with respect to these clusters will be high.

Hence, this approach lets us distinguish between common word clusters and topical clusters by the value of $\sigma_{u,c}$. The higher the standard deviation, the more likely it is that the cluster consists of common words. As the final scoring metric for the user profile, we propose to use the mean weight penalized by its variance; we used $\mu - 2\sigma$ as the final score in the examples below. This scoring metric can also be thought of as the lower bound of a confidence interval for the cluster affinity. Figures 1 and 2 show sample results of the users. Note how common-words clusters have high average affinity but also high standard deviation that drags them down in the final scoring and lets topical clusters come out on top.



#	μ	σ	Words
867	0.772	0.165	hours two-hour break minute half-hour five-minute two-hour ten-hour...
424	0.833	0.202	kissing call cry silent scream laughing nod dare restrain angry slam...
837	0.399	0.010	youtube blog net mail facebook player online yandex user tor ado...
366	0.396	0.042	associate attitude seems quite horoscope ideal religious face era...
413	0.406	0.080	feel glad remember worrying offended jealous inhale pity envy suffer autumn...
427	0.385	0.073	hijack bombing raid to steal loot bomb
798	0.385	0.080	uro missile air defense mine RL submarine Vaenga Red Banner Pacific Fleet...

Fig. 2. Sample user profile produced by resampling logistic regression.

3.4 Recommender Algorithm

Here, we present an actual recommender algorithm based on the user profiles mined as in Sect. 3.3. This serves as both a sample application for our user profiling system and as a way to evaluate our results numerically, by comparing it to baseline recommender algorithms.

We propose the following item-based algorithm to make recommendations based on a user profile in the form $\{(c, \mu_{u,c}, \sigma_{u,c}) | c \in C\}$:

(1) penalize the mean of a cluster's weight distribution with its variance, $w^\star_{u,c} = \mu_{u,c} - \alpha\sigma_{u,c}$, where α is a coefficient to be tuned for a specific system;
(2) predict the probabilities of likes according to the logistic regression model with modified weights, $p(\text{Like}|d, \boldsymbol{w}'_u) = \left(1 + \exp(\boldsymbol{v}_d^\top \boldsymbol{w}^\star_u)\right)^{-1}$;
(3) rank the items according to the predicted probability.

Note that this is a cold start algorithm for the items: it does not use an item's likes at all, only the likes of a user to construct his or her profile.

4 Evaluation

4.1 Experimental Setup

We have conducted experimental evaluation with a large dataset provided by the "Odnoklassniki" social network. For the experiment, we have chosen to use posts in groups (online communities) and likes provided by the users for these posts since a post in a group, as opposed to a post in a user's profile, is likely to be evaluated by many users with different backgrounds, and the users are more likely to like it based on its topic and content rather than the person who wrote it.

Thus, the dataset consists of texts of posts in the communities (documents) and lists of users who liked the posts. The basic dataset statistics are as follows:

– 286 K words in the vocabulary (after stemming and stop words removal);
– 14.3 M documents (group posts);
– 284.6 M total tokens in these documents;

As the user set U, we chose top 2000 users with most likes from a randomly sampled subset of users (so that we get users with a lot of likes but not outliers with huge number of likes that are most probably bots or very uncharacteristic users). We divided their likes into disjoint training and test sets; there were 16000 likes by these users in the training set and 4797 likes in the test set.

4.2 Experiments

We carried out our evaluation procedures on three algorithms: two baseline collaborative filtering algorithms and the new algorithm described above in Sect. 3.4.

1. User-based collaborative filtering: find k nearest neighbors for a user and recommend documents according to users' likes. Specifically, for each user u we build a list of k nearest neighbours $N(u)$ by cosine distance (via LSHForest) in the space of their vector representations in $\mathbb{R}^d$. Then we set the affinity between users as cosine distance between their vector representations:

$$w(u_1, u_2) = v_{u_1}^\top v_{u_2}, \quad \forall u_1, u_2 \in U.$$

   Documents are ordered by the following ranking function:

$$\text{rank}(u, d) = \frac{\sum_{u' \in N(u) \cap \text{Liked}(d)} w(u, u')}{\sum_{u' \in N(u)} w(u, u')},$$

   where $\text{Liked}(d)$ is the set of users who liked document d. Thus, we rank documents according to the weighted sum of representations of users who liked it.
2. Item-based collaborative filtering: find k nearest neighbors for a document and recommend documents similar to the ones a user liked. Specifically, for each $d \in D$ we build the set of k nearest neighbours $N(d)$ by cosine distance in the space of vector representations for the documents, compute similarities between the documents, $w(d_1, d_2) = v_{d_1}^\top v_{d_2}$, and rank documents as

$$\text{rank}(u, d) = \frac{\sum_{d' \in N(d) \cap \text{Like}(u)} w(d, d')}{\sum_{d' \in N(d)} w(d, d')},$$

   where $\text{Like}(u)$ is the set of documents user u liked.
3. Regression-based algorithm: recommend according to the negative-biased posterior distribution (Sect. 3.4); specifically, given the profile of each user $\{(\mu_{u,c}, \sigma_{u,c} | c \in C\}$, we rank documents according to

$$\text{rank}(u, d) = \frac{1}{1 + e^{-\sum_{c \in C} p(c|d) w^*_{u,c}}},$$

   where $w^*_{u,c} = \mu_{u,c} - \alpha \sigma_{u,c}$.

All users and documents vector representations are normalized before applying each of the algorithms above. Each of the evaluated recommender algorithms provides the ranking of documents for a given user. We build a set of all likes from test set and the same number of unliked documents for each user. It is expected that the liked documents will be ranked higher than others on average, which is a common ranking task. Hence, we used standard ranking evaluation metrics to evaluate the algorithms:

– NDCG (Normalized Discounted Cumulative Gain) is a unified metric of ranking quality [25]; the discounted cumulative gain is defined as

$$\text{DCG}_p = \sum_{i=1}^{p} \frac{\text{liked}_i}{\log_2(i+1)},$$

where $\text{liked}_i = 1$ iff item i in the ranked list is recommended correctly, and NDCG normalizes this value to the maximal possible:

$$\text{NDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p},$$

where the ideal DCG IDCG_p is the DCG of a ranked list with all correct items on top;

– Top1, Top5, and Top10 metrics show the share of liked documents at the first place, among the top five, and among the top ten recommendations respectively; these metrics are important for real-life recommender systems since an average user commonly views only a very small number of recommendations.

Results of our experimental evaluation are shown in Table 2. We see that the simple cold start recommender algorithm based on our user profiles performs virtually on par with collaborative filtering algorithms that actually take into account the likes already assigned to this item. These are very good results for a cold start algorithm; note, however, that actual recommendations of full-text items in the same system are not the only or even the main purpose of our approach: the ultimate goal would be to employ user profiles to make outside recommendations for other items with textual content or tags that could be related to the interest profile, such as targeted advertising.

Another way to demonstrate that the regression method learns new things about the users and items being recommended is to show its contribution into the performance of ensembles of rankers. We used the following blending method: first, we normalized the scores obtained by the methods in the blend ($Score_m$ for each ranking method m):

$$Score^m_{norm}(d) = \frac{Score^m(d) - \min_{doc} Score^m(doc)}{\max_{doc} Score^m(doc) - \min_{doc} Score^m(doc)}$$

for every document d and ranking method m, and then constructed the final scoring function as

$$Score_{blended}(d) = e^{\sum_m e^{Score^m_{norm}(d)} \alpha_m},$$

where α_m are blending weights to be found. We use hill climbing to tune parameters $\alpha_m \in [-1, 1]$, maximizing average NDCG for a separate validation set and finally

Table 2. Experimental results for the recommender algorithms.

	Algorithm	NDCG	Top1	Top5	Top10
1	User-based CF	0.7817	0.4557	1.9440	2.6557
2	Item-based CF	0.7904	0.4934	1.9636	2.6589
3	Regression-based cold start	0.7777	0.4852	1.8741	2.5960
4	User-based CF + regr	0.8089	0.5508	2.0130	2.6920
5	Item-based CF + regr	0.8043	0.5364	1.9834	2.6589

testing performance on a production set that constituted 20% of the values. Rows 4 and 5 of Table 2 show that the blends noticeably improved upon the performance of both our regression-based approach and classical collaborative filtering.

5 Conclusion

In this work, we have presented a new approach to user profiling based on logistic regression on randomly resampled subsets of items. This approach leads to readily interpretable user profiles in case of full-text recommender systems. Our experiments have shown that the simple cold start recommender algorithm based on user profiles produces results comparable to collaborative filtering approaches and can be blended with them for further improvement.

As future work, we envision further applications of this method with other methods for constructing the basic vectors. First, the proposed model may be improved by moving from word vectors to document vectors, e.g., the ones provided by the paragraph2vec models [26]. Full-text recommendations based on distributed word representations may be further improved, especially for the Russian language, by better training and design of custom deep models. On the other hand, for applications where a complete enough text dataset is available it might make more sense to use topic models rather than simple clustering of word vectors. The user profiling approach we have introduced remains applicable in all of these cases.

Acknowledgements. This work was supported by the "Recommendation Systems with Automated User Profiling" project sponsored by Samsung and the Government of the Russian Federation grant 14.Z50.31.0030. We thank Dmitry Bugaichenko and the "Odnoklassniki" social network for providing us with the social network dataset with texts of posts and user likes and Alexander Panchenko and Nikolay Arefyev for the trained *word2vec* model along with its Russian-language training data.

References

1. Webb, G.I., Pazzani, M.J., Billsus, D.: Machine learning for user modeling. User Model. User-Adap. Inter. **11**(1–2), 19–29 (2001)
2. Johnson, A., Taatgen, N.: User modeling. In: Handbook of Human Factors in Web Design. Lawrence Erlbaum, pp. 424–439 (2005)
3. Fischer, G.: User modeling in human–computer interaction. User Model. User-Adap. Inter. **11**(1–2), 65–86 (2001)
4. Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web: Methods and Strategies of Web Personalization. Springer, Heidelberg (2007)
5. Bjorkoy, O.: User modeling on the web: an exploratory review of recommendation systems. PhD thesis, NTNU Trondheim (2010)
6. Lops, P., Gemmis, M.D., Semeraro, G., Lops, P., Gemmis, M.D., Semeraro, G.: Chapter 3 content-based recommender systems: state of the art and trends
7. Pazzani, M.J., Billsus, D.: The Adaptive Web. Springer, Heidelberg (2007)
8. Pazzani, M., Billsus, D.: Learning and revising user profiles: the identification ofinteresting web sites. Mach. Learn. **27**(3), 313–331 (1997)
9. Middleton, S.E., Shadbolt, N.R., De Roure, D.C.: Ontological user profiling in recommender systems. ACM Trans. Inf. Syst. **22**(1), 54–88 (2004)

10. Billsus, D., Pazzani, M.J.: User modeling for adaptive news access. User Model. User-Adap. Inter. **10**(2–3), 147–180 (2000)
11. Cohen, W.W.: Fast effective rule induction. In: 12th International Conference on Machine Learning (ML95), pp. 115–123 (1995)
12. Basu, C., Hirsh, H., Cohen, W.: Recommendation as classification: using social and content-based information in recommendation. In: Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, AAAI 1998/IAAI 1998, pp. 714–720, Menlo Park, CA, USA. AAAI (1998)
13. Al-Rfou, R., Perozzi, B., Skiena, S.: Polyglot: distributed word representations for multilingual NLP. In: Proceedings of the 17th Conference on Computational Natural Language Learning, Sofia, Bulgaria, ACL, pp. 183–192, August 2013
14. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, Association for Computational Linguistics, pp. 1532–1543, October 2014
15. Panchenko, A., Loukachevitch, N., Ustalov, D., Paperno, D., Meyer, C.M., Konstantinova, N.: RUSSE: the first workshop on Russian semantic similarity. In: Proceedings of the International Conference on Computational Linguistics and Intellectual Technologies (Dialogue), pp. 89–105, May 2015
16. Kumar, B.V., Kotsia, I., Patras, I.: Max-margin non-negative matrix factorization. Image Vision Comput. **30**(45), 279–291 (2012)
17. Arefyev, N., Panchenko, A., Lukanin, A., Lesota, O., Romanov, P.: Evaluating three corpus-based semantic similarity systems for Russian. In: Proceedings of International Conference on Computational Linguistics Dialogue (2015, to appear)
18. Vorontsov, K., Frei, O., Apishev, M., Romov, P., Suvorova, M., Yanina, A.: Non-Bayesian additive regularization for multimodal topic modeling of large collections. In: Proceedings of the 2015 Workshop on Topic Models: Post-Processing and Applications, TM 2015, pp. 29–37, New York, NY, USA. ACM (2015)
19. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**(4–5), 993–1022 (2003)
20. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. SIGMOD Rec. **25**(2), 103–114 (1996)
21. Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. Data Min. Knowl. Discov. **2**(2), 169–194 (1998)
22. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
23. Sculley, D.: Web-scale k-means clustering. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, pp. 1177–1178, New York, NY, USA. ACM(2010)
24. Arthur, D., Vassilvitskii, S.: K-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, pp. 1027–1035, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics (2007)
25. Jarvelin, K., Kekalainen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. **20**(4), 422–446 (2002)
26. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Jebara, T., Xing, E.P., (eds.) Proceedings of the 31st International Conference on Machine Learning, JMLR Workshop and Conference Proceedings, pp. 1188–1196 (2014)