A question answer-based context-aware recommendation method with semantic model

Jorge Castro[a,b,*], Jie Lu[b], Guangquan Zhang[b], Luis Martínez[c]

[a]*Department of Computer Science and Artificial Intelligence, University of Granada, Granada (Spain)*
[b]*School of Software, University of Technology Sydney, Sydney (Australia)*
[c]*Computer Science Department, University of Jaén, Jaén (Spain)*

Abstract

Content-Based recommender systems (CB) filter relevant items to users in overloaded search spaces using information about their preferences. However, classical CB scheme is mainly based on matching between items descriptions and user profile, without considering that context may influence user preferences. Therefore, it cannot achieve high accuracy on user preference prediction. This paper aims to handle this issue to improve quality of recommendation taking contextual information as the trend in current collaborative interest, where a feed of status updates can be analyzed to model the context. It proposes a novel CA-CB approach that recommends question/answer items and introduces context awareness based on topic detection within collaborative interest. A case study and related experiments show that the context integration significantly benefits recommendation.

Keywords: `content-based recommender system`, `context-aware recommendation`, `user profile contextualization`, `map-reduce`, `data streaming`

1. Introduction

The increasing amount of information available in World Wide Web scenarios, such as e-commerce, affects users' satisfaction when they search for items that meet their interests. This situation originates that users need to put significant effort for finding

*Corresponding author

Email addresses: `jcastro@decsai.ugr.es` (Jorge Castro), `jie.lu@uts.edu.au` (Jie Lu), `guangquan.zhang@uts.edu.au` (Guangquan Zhang), `martin@ujaen.es` (Luis Martínez)

relevant pieces of information for them. In some scenarios it might not be possible for users to explore information items in order to select the most suitable one. Hence, the *information overload* problem impacts users satisfaction. Recommender systems have been a powerful tool for alleviating information overload in large search spaces. Recommender systems have been proved to be successful in several domains, such as e-business [26], e-learning [48, 50], e-tourism [5, 34], e-commerce [43], web pages [32, 49] and financial investment [30], among others.

Several approaches have been explored for alleviating information overload problem with recommender systems. The most widespread ones are collaborative filtering [23] and Content Based (CB) [15]. The main difference between them is that collaborative filtering focuses on users' interaction with items, i.e., user preferences, while CB focuses on the analysis of items descriptions, i.e., item content. Therefore, the performance of these recommendation approaches is subject to the quality and amount of available information of both types. In addition to these successful strategies, other approaches have been proposed, such as knowledge-based recommender systems that focus on employing expert information over the recommendation domain through ontologies [33], among other [14, 48], or social network recommender systems that use links between users to improve the recommendation [18]. Recent research lines also focus on integrating contextual information [3] or providing recommendations targeted to groups of users[12, 28]. In this paper, we focus on recommendation of question answering (QA) items with content-based approach and contextual information integration [24].

Within CB we distinguish two kinds: (i) based on item features, and (ii) based on items descriptions. In this paper we focus on the latter, given that QA items have a strong component of textual information for both explaining the question and answering it [20]. Although CB suffer from user cold start because they need some input from user preferences and lack of diversity in recommendations [8], CB have demonstrated their utility when new items are introduced in the system, i.e., in scenarios with strong item cold-start [4]. This feature makes the application of CB approaches interesting in domains where new items are constantly introduced, such as web pages or news. In this direction, QA recommendation shares the features to apply CB with textual descriptions, hence, we focus on them.

In the QA domain recommendation scenario there are several proposals for recommendation with CB approaches. Shao et al.[46] propose to apply Latent Dirichlet Allocation to label questions in a latent semantic feature category and find the most suitable answerers. In this direction, Zheng et al. [52] combine trust-based analysis of answerers with content analysis. While these approaches aim to reduce the answer time searching for adequate answerer for new posted questions, there are other recommender systems that aim to expand users' knowledge recommending already answered questions. Odiete et al. [35] analyze users' preferences and build a graph of expertise used to find gaps in their knowledge and suggest relevant questions.

Within QA domain recommendation, it is interesting to focus on recommending answered questions that are in the target user's area of interest, and that are also relevant regarding the collaborative interest. Therefore, it is interesting to explore Context-Aware CB (CA-CB), which integrate contextual information to the content-based recommendation. De Pessemier et al. [16] consider recommendations in mobile devices as very suitable to integrate context-awareness, and uses devices sensors and time of the day to deliver contextualized news recommendations. In QA recommendation, Sea-Hawk [40] and Prompter [39] provide CA-CB that support programmers to complete issues and bugs using query completion and recommends StackOverflow questions, where the context is the specific part of the source code from which the recommendations are requested. In this direction, Libra [41] also integrates recommendations but it also considers, in addition to context extracted from the Integrated Development Environment, resources opened by the user such as URLs or documents to better understand his/her context. Other works consider collaborative interest as current buzzwords to deliver currently relevant recommendations in e-commerce scenarios [38]. As it can be seen, there is no previous approach that focuses on context-aware recommendation regarding collaborative interest in the QA domain.

In this paper, we propose a novel CA-CB approach that recommends QA items and introduces context awareness based on topic detection within collaborative interest. Recent researches [22] have highlighted the immediacy of microblogging services such as Twitter, where users share short sentences or fragments of news. In this proposal, the context is extracted from microblogging systems to characterize current trends in collaborative interest. The usage of such a context mainly helps recommend answers related to topics of interest and indirectly overcomes the overfitting problem. However, the context extracted this way is often noisy or several topics are mixed. With this regard, we propose to cluster context to identify the topics that are being discussed, and after that the context topic most suitable to target user's preferences is selected to build a contextualized user profile that combines preferences and context. This way, the proposal provides contextualized recommendations that are also adjusted to users' individual interests. Moreover, QA domain has data in high volume and microblogging systems generate data at high rate. In this regard MapReduce framework, has been proved to be effective in those scenarios [27, 31], therefore, our proposal applies it.

The contributions of this study are:

- A suitable way to study status updates in the QA recommendation scenario to provide recommendations tailored to both the user preferences and current collaborative interest.
- Introduction of personalised contextualization of user preference profiles to better integrate collaborative interest context in the recommendation.
- A proposal that applies semantic analysis of QA domain, deals with mixture of topics in collaborative interest and provides personalised context-awareness in the recommendation.
- A case study and experimentation that validates the proposal and determines that integration of contextual information extracted from collaborative interest improves QA recommendation.

The remaining of this paper is structured as follows. Section 2 provides a background of CB, CA-CB and the MapReduce approach. Section 3 introduces in further detail our proposal of CA-CB for QA recommendation. Section 4 shows a case study performed to evaluate the proposal and discuss the findings. Finally, Section 5 concludes the paper.

2. Preliminaries

This section provides the required background for the current research, including basics about CB, related works in content based recommendation with context awareness and basics in MapReduce and Spark.

2.1. *Content-based recommender systems.*

An accurate definition of recommender system, given by R. Burke [10], is *"any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options"*. Within recommender systems, various techniques can be distinguished based on the knowledge source [16]: demographic, knowledge-based, community-based, content-based, collaborative, and hybrid recommendations. Among them, we focus on CB.

The various CB approaches can be classified regarding the item representation. Here we focus on those CB that use a vector space modeling to represent items. With this regard there are CB with (i) feature-based representations, and (ii) free-text representation.

In feature-based representation, items are usually stored in a database table where rows are items and columns are the item features (also called fields or properties) and each item might have a different value for each of these features. The recommender system learns a model that uses such information to approximate the rating function. There are proposals based on weighting features importance regarding the user's ratings using collaborative information [47] or using entropy and dependence between items' features and user's ratings [11]. Other approaches learn the rating function using linear regression [16].

In free-text representation there is a natural language piece of text that describes the item, such as movie synopsis or that is part of the item itself, such as the content of a web page or a news article. Such a kind of unstructured data provides information about the item (called document in these systems), however, they also have the complexity of dealing with natural language due to polysemous words and synonyms.

The TFIDF approach [19] is usually applied when dealing with free-text representation items. In TFIDF, the unstructured data is converted in structured data stemming words [42] to keep their root. This process reduces the number of components of documents unifying words such as computer, compute and computing, which are different forms that share meaning. After that, for each document, a vector of weights of each term is generated multyplying the $tf_{t,d}$ by the idf_t [25], to consider the importance of the term on the document:

$$profile_d^{tfidf} = \{tf_{t,d} * idf_t \quad s.t. \quad t \in d\} \tag{1}$$

where $tf_{t,d}$ is the number of occurrences of term t in document d, N is the set of all documents and N_t is the set of documents that contain the term t at least once.

$$idf_t = -\log\left(\frac{|N|}{|N_t|}\right) \tag{2}$$

At this point the system contains a vector space model of items. User profiles can be generated aggregating the profiles of the items that they liked in the past [47]. The recommendation is computed comparing user profiles with item profiles with the cosine correlation and the closest ones are recommended.

While TFIDF method is effective, it cannot deal with polisemy or synonym words. In order to overcome this issue, Latent Semantic Analysis (LSA) is applied [15]. In LSA, the term-document matrix is factorized with Singular Value Decomposition (SVD) to reduce it to orthogonal dimensions and keep the f most relevant singular values (see Figure 1).

$$TFIDF_{(|D|\times|T|)} = U_{(|D|\times f)} * s_{(f)} * V^t_{(f\times|W|)} \tag{3}$$

This way, a reduced feature space is defined, which properly manages noise and redundancy of terms. User profiles are generated from this feature-space definition through a linear combination of document profiles that they liked [7]. Then, recommendations are generated comparing user and document profiles with cosine correlation coefficient.

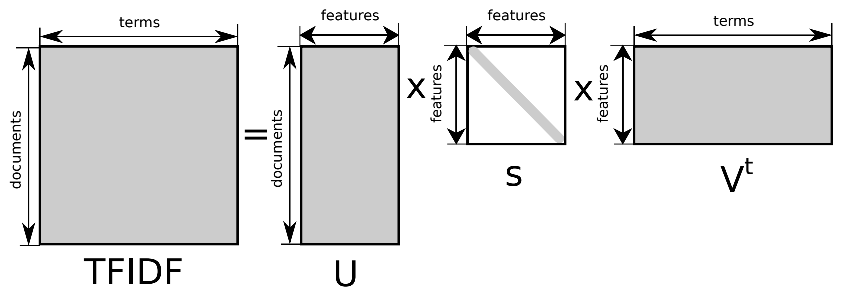

Figure 1: Decomposition of TFIDF matrix with singular Value decomposition. Note that *s* is a diagonal matrix with the singular values sorted in descending order.

2.2. Context-aware recommender systems.

In addition to the information traditionally used by recommender systems, as noted by R.Burke [10], other sources of information can be considered in the recommendation, such as the context in which the recommendation is received by users. F. Ricci [44] stated that the conditions or circumstances in which the recommendation is delivered significantly affect users' decision behavior. Therefore, the consideration of users' context is key to provide interesting recommendations.

With this regard, the various context-aware recommendation approaches can be classified into three classes [2]:

- Pre-filtering: The system selects and uses only the feedback gathered in the same context in which the recommendation is delivered to the user.
- Post-filtering: The recommendations are generated first without considering contextual information. After that, the item predictions are modified regarding the specific context of the users, possibly filtering out some items.
- Contextual modeling: The contextual information is directly integrated in the model that is used to recommend.

In pre-filtering, the approach is to filter out the information that was not not gathered in the current context. In traditional recommender systems the information is viewed as a function $R : User \times Item \rightarrow Rating$ that the recommender system tries to approximate. In CA, the information can be viewed as a three dimensional cube

$User \times Item \times Context \rightarrow Rating$. Contextual pre-filtering selects only the information relative to the context, hence, it tries to approximate function $R_{context} : User \times Item \rightarrow Ratings_{context}$ This way, they only consider ratings generated in the context in which the recommendation is delivered to the user. Contextual pre-filtering is a simple and effective approach, but it is affected by data sparsity when there is not enough information generated in all contexts to provide accurate recommendations. Some researchers have tried to overcome this issue through context-generalization [2] when the information available in the current context is not enough and include information from the broader context.

In post-filtering, the recommendations are first computed overlooking the contextual information, as in traditional recommender systems. After this initial step, recommendations are adjusted to the current context either removing irrelevant items or through a weighting function that updates predictions regarding the suitability of items to target user's context. A previous work [37] evaluated them in the same scenarios and compared pre- and post-filtering approaches. It determined that neither pre-filtering nor post-filtering completely dominates the other and a study to determine the best approach is needed in each specific case.

Previous approaches try to reduce the CA problem to a two dimensional one that can be solved with traditional recommender systems. This is not the case in contextual modeling, in which contextual information is directly integrated in the recommendation model to recommend. With this regard, researchers have explored heuristic-based [36], probabilistic [1], or matrix factorization [6] contextual modeling approaches.

2.3. MapReduce and Spark

MapReduce [17] is a processing tool to manage data in an scalable way. It was originally designed by Google in 2003 aiming to support the most powerful search-engine on the internet, but later it was also applied for parallelization in general purpose applications. In MapReduce (see Figure 2), two main operations are defined by the user: *Map* and *Reduce*. The *Map* operation takes an input, which is given as a set of <key,value> pairs, and produces an intermediate set of <key,value> pairs that can be of different type and are defined by the user. An intermediate phase merges all pairs that have the same key and redistributes the work. The *Reduce* operation combines pairs by key to produce a smaller pair set.

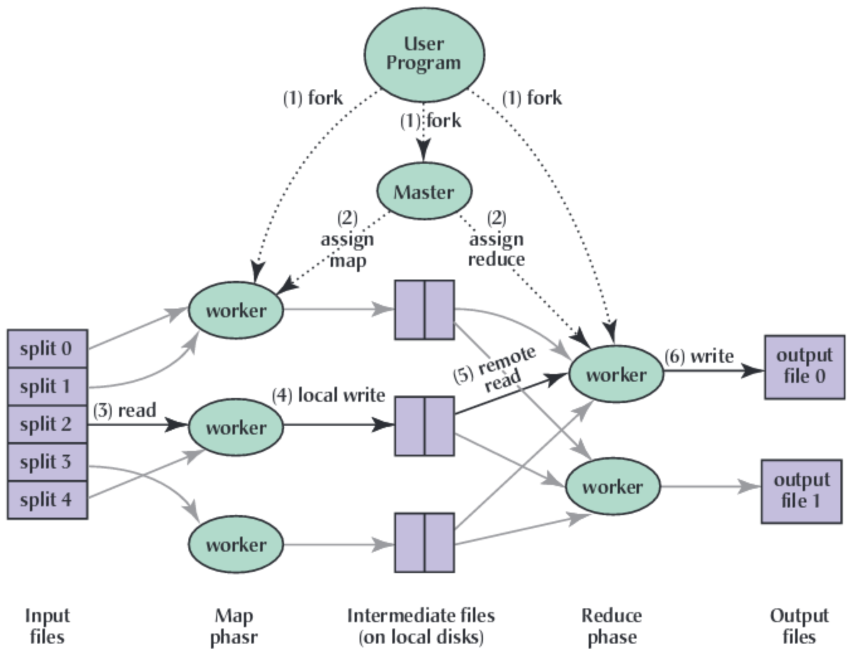

Figure 2: Scheme of the MapReduce approach [17].

There are several solutions that apply the MapReduce approach. Among them, Apache Spark [51], which was introduced as part of the Hadoop Ecosystem, offers to the user a set of in-memory primitives that complement the MapReduce ones and that is suitable for iterative tasks. It is based on Resilient Distributed Datasets (RDDs), a structure that stores data in such a way that later computations can be easily parallelized in distributed machines. RDDs allow to cache or redistribute intermediate results, which enables the design of data processing pipelines.

Within Spark we use two libraries: MLlib and Spark streaming. MLlib is a scalable machine learning library [29] that was built to take advantage of Spark suitability for iterative tasks and provides several machine learning techniques for classification, optimization, and data pre-processing, among others. Specifically, we use the tools that MLlib provides for regression and clustering. Spark Streaming [13] provides an scalable way to manage data produced at high rates, which allow us to handle the data provided by microblogging systems and compute the context model.

3. Semantic model for recommending questions with context awareness based on topic detection in collaborative interest

Here, a novel proposal, LSAContextCluster, for recommendation based on CA-CB is introduced. Recommendations might need to be targeted to specific contexts, e.g., when a system delivers recommendations of answers in the history domain and currently people are posting about Colombus Day, then the system should promote answers related to the discovery of the Americas. In this kind of cases, it is possible to modify user profiles to include contextual information in such a way that later recommendations are both targeted to user preferences and current context.

The proposed model fits into the CA approach of contextual modeling, because it integrates contextual information in the model built by the recommender system. The general scheme of the proposal, LSAContextCluster, is depicted in Figure 3, and it is composed of five phases:

(i) QA domain semantic analysis: It applies LSA to reduce the dimensionality of the term-document matrix.

(ii) Build user preference profile: It analyses users' preferences and generates a profile for each of them based on the profiles of the document he/she liked in the past.

(iii) Build context model: It analyses the context, which consists on a stream of status updates within a given time frame, applies clustering to separate the various topics that the context contains, and generates feature-space profiles for each context topic.

(iv) Contextualize user profiles: It selects the context topic that is most suitable to target user's preferences and combines the preference-based user profile with the context topic profile to generate the contextualized user profile.

(v) Prediction: It compares the document profiles and the contextualized user profile to recommend.

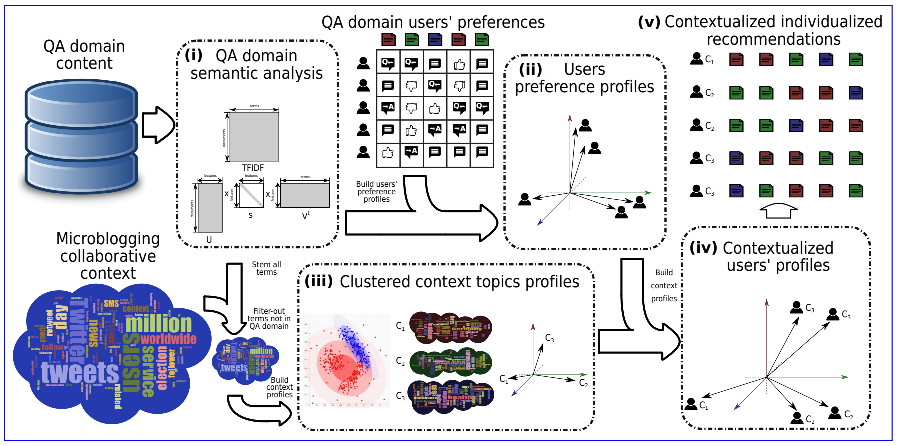
QA domain content
(i) QA domain semantic analysis
TFIDF
U
s
V^t
QA domain users' preferences
Build users' preference profiles
(ii) Users preference profiles
(v) Contextualized individualized recommendations
Microblogging collaborative context
Stem all terms
Filter-out terms not in QA domain
Build context profiles
(iii) Clustered context topics profiles
Build context profiles
(iv) Contextualized users' profiles
C1
C2
C3

Figure 3: General scheme of the proposal.

3.1. QA domain semantic analysis

Our proposal assumes that the QA dataset contains textual information of the question and their related answers. In this proposal we consider the questions together with all their answers as the document, and the words used in their text as the terms. The terms are stemmed using the Porter Stemmer algorithm [42]. Once terms are stemmed, the TFIDF document profiles $profile_d^{TFIDF}$ are built according to Eq. 1.

Once the TFIDF document profiles are built, LSA is performed to reduce the dimensionality of the matrix. LSA is proven to be effective through the description of both document and terms in a feature space with a reduced number of features. Therefore, the aim of this step is to decompose the initial word-document matrix in a word-features matrix U, a singular value vector s, and a document-features matrix V (see Eq. 3).

An approximated factorization of the TFIDF matrix is performed with Singular Value Decomposition, which allows to reduce the dimensionality of the original matrix keeping the f most relevant singular values of the original matrix. Hence, we obtain the profile of both terms and documents in the feature space, which compose the QA domain semantic model:

$$profile_d^{LSA} = \{u_{t,1}, \dots, u_{t,f}\} \tag{4}$$

Table 1: Users' preferences over items, the rating matrix.

	d_1	$\dots$	d_k	$\dots$	d_n
u_1	r_{u_1,d_1}	$\cdots$	r_{u_1,d_k}	$\cdots$	r_{u_1,d_n}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
u_j	r_{u_j,d_1}	$\cdots$	r_{u_j,d_k}	$\cdots$	r_{u_j,d_n}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
u_m	r_{u_m,d_1}	$\cdots$	r_{u_m,d_k}	$\cdots$	r_{u_m,d_n}

$$profile_t^{LSA} = \{v_{t,1}, \dots, v_{t,f}\} \tag{5}$$

3.2. Building user preference profile

At this point LSAContextCluster has built a model with terms and documents profiles. In order to provide personalized recommendations to users, it is needed to build user profiles in the same feature-space. LSAContextCluster holds a unary matrix that states whether a given user has expressed interest in another document (see Table 1) either creating, commenting, or voting it. In this table, the set of documents that user u has expressed interest in is defined as:

$$R_u = \{d \quad s.t. \quad r_{u,d} \in R\} \tag{6}$$

This way, the user's profile is built upon the profiles of the documents that belong to R_u, and describes the user's preferences in terms of the latent space:

$$profile_u^{LSA} = \sum_{d \in R_u} profile_d^{LSA} = \{\sum_{d \in R_u} profile_{d,1}^{LSA}, \dots, \sum_{d \in R_u} profile_{d,f}^{LSA}\} \tag{7}$$

3.3. Context model building

To include contextual information in the recommendation process, it is needed to build the context model. In this proposal, we aim to promote questions that are relevant regarding current happenings. With this regard, our proposal uses status updates from microblogging services, such as Twitter, as the source of collaborative interest. Formally, a status update consists of a free text input generated by a user with certain timestamp, among other meta-data. Analyzing these status updates, LSAContextCluster generates a model of the context that is later used to modify the user profile. The scheme of the context model building phase is depicted in Fig. 4.

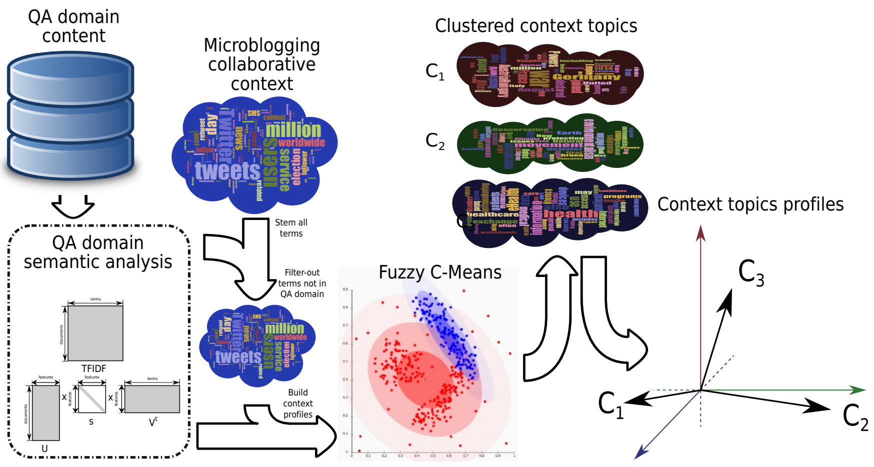

Figure 4: Context model building phase.

The context is composed of the status updates that were generated in a given time window, which is set to 24 hours in this proposal, although it could be modified to adjust the sensitivity of the context model. First, all terms of the status updates of the current context are stemmed. After that, from all the terms that the context contains, LSAContextCluster filters out the terms that do not appear in the QA semantic model generated in phase one (see Section 3.1).

Given that the context is composed of several status updates, there can be a mixture of topics. To determine the context topics, the proposal performs a fuzzy clustering of the terms used in the context. Hence, fuzzy c-means clustering algorithm [9] groups the terms using their feature vector $profile_t^{LSA}$ as the term definition. The distance among terms used in the clustering is based on cosine correlation coefficient. The result is a set of clusters where each cluster c_i defines a context topic.

Once the terms of current context are grouped in context topics, the proposal generates a context profile for each topic combining the profiles of the terms that are included in each cluster. Therefore, LSAContextCluster builds a profile for each context topic using the feature representation of each term from the QA domain (see Eq 8). At the end of this phase, LSAContextCluster has generated a model of the context composed of several context profiles, one for each context topic detected by the clustering.

$$profile_{c_i}^{LSA} = \sum_{t \in c_i} profile_t^{LSA} = \{\sum_{t \in c_i} profile_{t,1}^{LSA}, \ldots, \sum_{t \in c_i} profile_{t,f}^{LSA}\} \tag{8}$$

3.4. User profile contextualization

In this step, the target user's preference profile is combined with the context model to provide contextualized personalized recommendations. To do so in a personalized way, from all the context topic profiles that the context model contains, LSAContextCluster selects the most similar to the user's preference profile. This way, the context topic c_i that has a greater cosine coefficient with the target user preferences is used to modify his/her profile, hence, the contextualization of user profiles is personalized to user preferences.

$$\underset{c_i}{\operatorname{argmax}} \quad cosine(profile_u^{LSA}, profile_{c_i}^{LSA}) \tag{9}$$

After this selection, the profile of the selected context topic c_i and the user's preference profile are combined to obtain the contextualized user profile. With this regard, the convex combination is applied, which allows to perform a weighted combination regulated by α parameter – the greater its value, the more importance of the user's preference profile over the profile of the selected context topic in the contextualized user profile.

$$profile_{C,u}^{LSA} = \alpha * profile_u^{LSA} + (1 - \alpha) * profile_{c_i}^{LSA} \tag{10}$$

3.5. Prediction

Once we have the contextualized user profile, we can produce a prediction of the suitability for a given item regarding the profile. The recommendation is a list of documents sorted by $p_{u,d}$:

$$p_{u,d,c} = profile_{C,u}^{LSA} * s * profile_d^{LSA} \tag{11}$$

4. Case study and experiment

To evaluate the proposal, we performed an experiment that simulates the recommendation of QA items in various contexts. The remainder of the section is structured as follows. First, the settings of the experiment are described. The datasets and methods for processing them are then detailed. After that, the evaluation measures are commented. Lastly, the results are analyzed.

4.1. Experimental procedure

In these experiments we compared several CB approaches based on LSA with contextual information. In order to do the experiment, the procedure proposed by Sarwar et al. [45] was performed with modifications to consider contextual information in the experiment:

- Split the dataset in training and test.
- Build the model with training data.
- Build the profile of each user including contextual information if applicable.
- Recommend to each user based on their profile and the model.
- Evaluate recommendations with the test set.

This procedure was repeated 20 times and 5-cross fold validation was used to split the data in training and test sets. Moreover, various contexts were considered, which are detailed in Section 4.3.

4.2. Methods compared

The baseline method to compare with was the LSA method without contextual information. For the sake of fair comparison, the number of features was fixed in all models, and 30 features were considered in LSA.

We compared several ways for integrating contextual information in QA recommendation. Here, we show the results of three ways to characterize the context:

- No clustering (LSAContext): The words are not separated in clusters, therefore the context profile is unique. There is a single profile of the context that is built combining the profiles of the words that are included in the context.

$$profile_C = \sum_{t \in C} * profile_t \tag{12}$$

- Weighted by membership (LSAContextClusterFuzzy): The cluster profiles are built combining the feature vector of each word weighted by the membership value of the word to the cluster:

$$profile_{c_i} = \sum_{t \in c_i} \mu_{t,c_i} * profile_t \tag{13}$$

  where μ_{t,c_i} is the membership of term t to cluster c_i.

- Max membership (LSAContextClusterMax): The words are used only in the cluster to whom they have the highest membership value:

$$profile_{c_i} = \sum_{t \in T} \mu_{t,c_i}^{max} * profile_t \tag{14}$$

  where μ_{t,c_i}^{max} is one if μ_{t,c_i} is the maximum membership across clusters, and zero otherwise.

Moreover, in order to adjust the weight of preference profile over context profile, the methods compared have parameter α. In the experiment we explored several values for it, here, to make the results clearer, we show only $\alpha \in [0.90, 1.00]$ with increments of 0.01.

4.3. Datasets

In the experiment there are two sources of data: The QA domain and the contextual information.

The QA domain used is the StackExchange dataset[1]. This dataset consists of the

[1]http://data.stackexchange.com/

Table 2: Main features of the QA domain datasets used in the case study.

	3dprinting	academia	ai	android	apple	history
Users	4025	48448	3158	119810	145011	13433
Questions	597	57967	421	41423	77978	6127
Answers	1135	16737	749	49985	115643	12212
Comments	2754	40319	1165	123291	242238	53510
Votes	7860	138416	5323	359710	669305	162809
Ratings	2458	119082	1461	109644	241838	38082
Sparsity	0.99898	0.99996	0.99890	0.99998	0.99998	0.99954

database dump of each site in the stackexchange ecosystem. Across them, we focus on 3dprinting, a StackExchange site devoted to it. Some stats are detailed in Table 2.

In this experimental setup, the results are reported per StackExchange site. Therefore, we consider each site as a different dataset. Given that the aim of the system is to provide users with pieces of information that help explain the context and also consider their preferences.

Regarding the contextual dataset, a set of interesting keywords is defined based on the aim of the proposal for contextualizing recommendations. Given that the proposal focuses on selecting currently hot topics, we have selected the terms *news*, *current* and *situation*. From these seed terms, we extracted a dataset of tweets that contain any of these words from Twitter. The stats of the dataset extracted is depicted in Figure 5.

4.4. Evaluation Measures

Usually, measures to evaluate the prediction errors in terms of rating deviation are used. However, the methods being compared do not provide a rating prediction, but a value that expresses the suitability of items regarding the user profile. Therefore, the measures that can be used are information retrieval ones, such as precision and recall. Researchers have remarked that, although they are useful, they are not sensible to the sorting of the items that the recommender systems does [21]. In order to consider the quality of the sorting, the NDCG is used:

$$DCG_u = \sum_{k=1}^{N} \frac{r_{u,recom_{u,k}}}{log_2(k+1)} \tag{15}$$

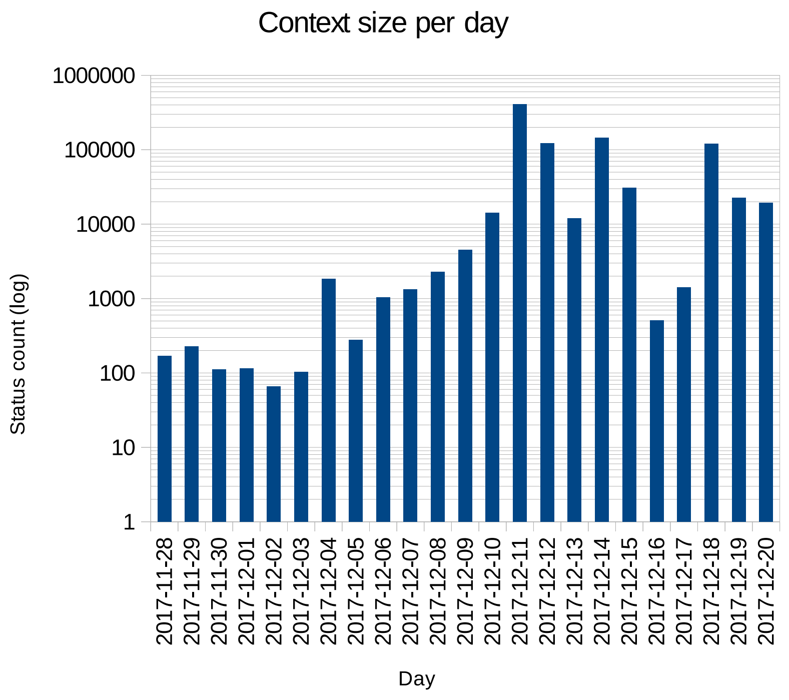

Figure 5: Contextual dataset used in the experiment, where each day has a different status count.

where $recom_{u,k} \in I$ is the item recommended to user u in k position.

$$NDCG = \frac{DCG}{DCG_{perfect}} \tag{16}$$

where $DCG_{perfect}$ is a perfect sorting of the items, i.e., the list of items sorted by their value in the test set.

4.5. Results

In this section, the results obtained for the different approaches compared are shown and analyzed to evaluate the performance of the proposal. Figures 6, 7, and 8 show the results of the 3 techniques compared in the 3dprinting QA dataset. The three figures have the same scale and, in X axis, alpha parameter is shown. The series denote the context, hence its position shows the results of the proposal with the corresponding alpha value for the day.

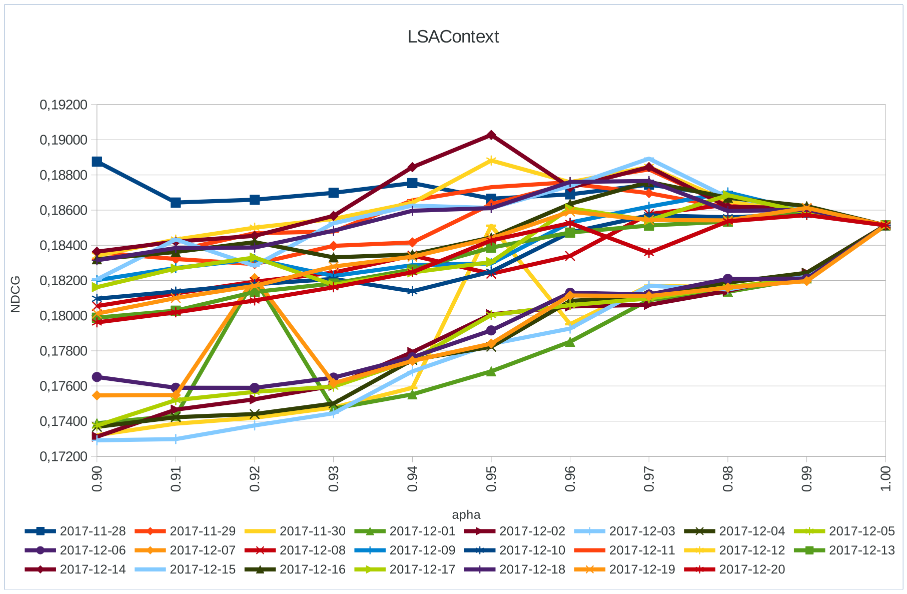

Figure 6: Results of the proposal to manage context without clustering

In Figure 6 it can be noticed that, although the proposal improves the results of LSA in some days (contexts), the improvement does not compensates for the decay in performance in other days. Focusing in the context of 2017-11-28, LSAContext improves the results of LSA for all alpha values.

Figure 7 shows that LSAContextFuzzy improves the results of LSA. It obtains better results than LSAContext, given that the results are distributed higher than those of the LSAContext approach. If we focus on the results on single days, it can be noticed that LSAContextFuzzy improves greatly for the context of day 2017-11-28. However, there is a major decay in three contexts: 2017-11-29, 2017-11-30 and 2017-12-15 in which the proposal does not even reach the value of the LSA approach. If we focus specifically on the day 2017-12-12, there is a decay for $\alpha \in [0.90, 0.95]$ but it improves the results of LSA for $\alpha \in [0.96, 0.99]$.

Figure 8 shows that LSAContextClustering improves the results of LSA for most of the contexts explored. It consistently obtains better results than LSA, although the improvements obtained in some contexts is canceled with worst results in other contexts.

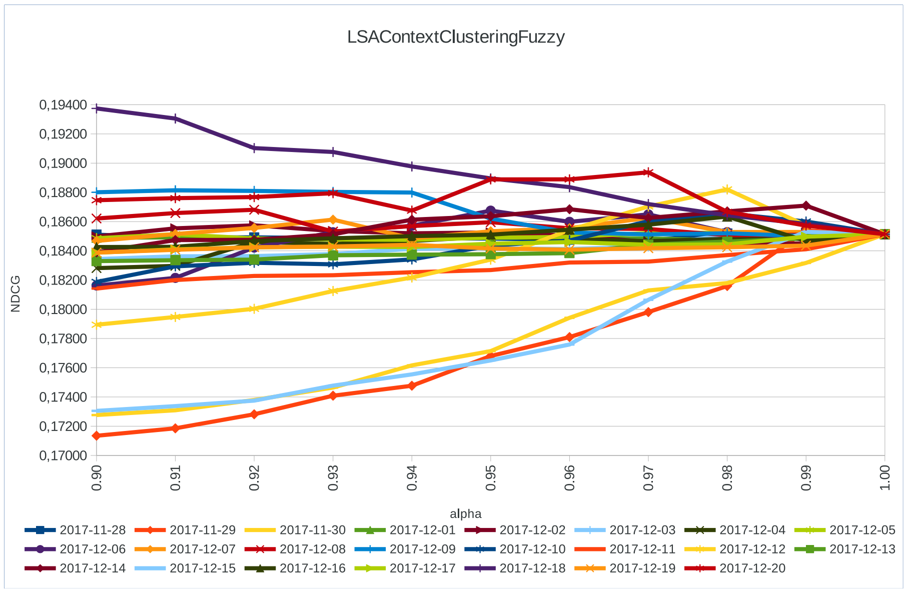

Figure 7: Results of the proposal to manage context with fuzzy membership.

Figure 9 compares the results of each proposal with the best alpha. Here LSA has no variability across days because it does not considers context. LSAContext has a great variability in performance across days, however, the better results obtained from 2017-12-08 onwards are canceled by the low performance from 2017-11-30 to 2017-12-07. LSAContextClusteringMax and LSAContextClusteringFuzzy performances did not dropped drastically in certain contexts as it happened for LSAContext. Instead, they show ups and downs in the contexts. It is worth to notice that LSAContextClustering-Max obtained greater peaks than LSAContextClusteringFuzzy.

Figure 10 summarizes the results of the three proposals as compared to the LSA approach. It shows that LSAContextFuzzy and LSAContext, although they obtain better results than LSA in some contexts, in average they do not provide improvement. In the case of LSAContextClustering, it overcomes the results of all the remaining approaches for $\alpha \in [0.90, 0.97]$. For $\alpha = 0.94$ it reached the maximum average NDCG across all contexts explored, hence this value is the best one in this QA domain.

The best approaches of the compared ones is the LSAContextClustering with $\alpha = 0.94$. This value has been optimized for the 3dprinting QA dataset, hence, for other

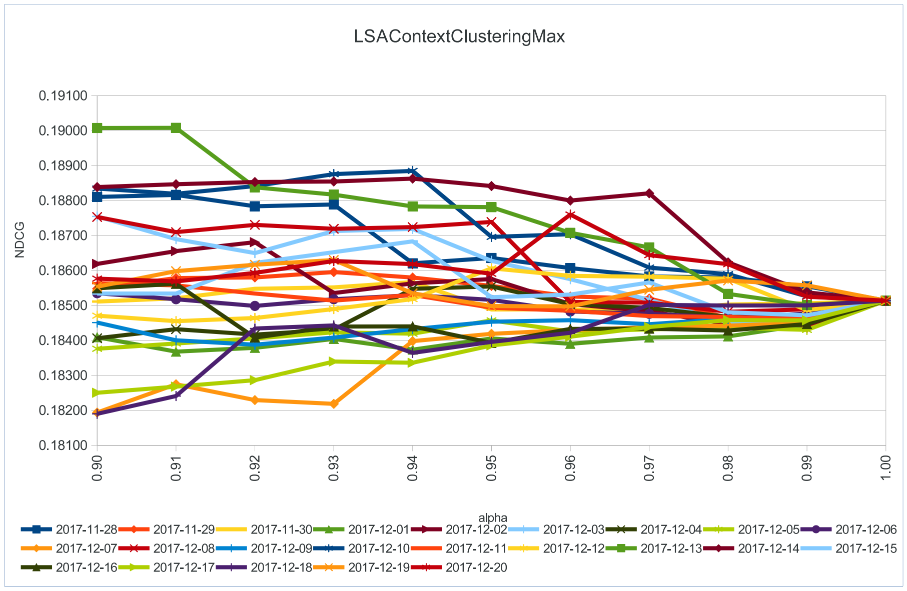

Figure 8: Results of the proposal to manage context with max membership.

domains it needs to be adjusted. This parameter provides the LSAContextClustering with flexibility to adapt to different QA domains.

5. Conclusions and further study

In this paper, we have explored the application of contextual information in the QA domain recommendation. LSAContextClustering first builds the LSA model associated to the QA domain. After that, it builds the user profile combining the QA profiles with the user preferences. In parallel, it builds the profiles of the context, which is separated in a number of clusters and a context profile is built for each of them. The following step is to combine the user profile with the context profile that is more close to their preferences, which is achieved computing the cosine coefficient between the profiles. This combined profile allows the system to know user preferences and also consider contextual information in the recommendation.

We performed a case study to compare various configurations for the proposed approach. It shows that all improvements done provide better results as compared to the baseline method (LSA). We found out that the best way to generate each context

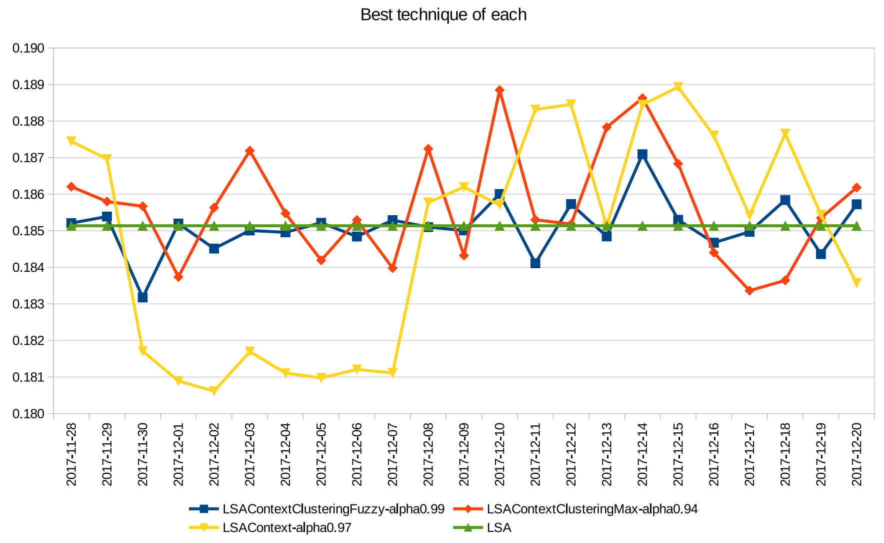

Figure 9: Results of the proposal to manage context with fuzzy membership.

cluster profile is to select only the words whose membership value is the highest across clusters in the explored QA domain.

In this scenario, contextual information is a key source of information to provide users with relevant recommendations that allow them to better understand the current scenario. The provided system is a relevant tool in the completion of user knowledge through the recommendation of QA items.

Acknowledgements: This research work was partially supported by the Research Project TIN-2015-66524-P, the Spanish FPU fellowship (FPU13/01151).

Bibliography

[1] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, 2005.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender

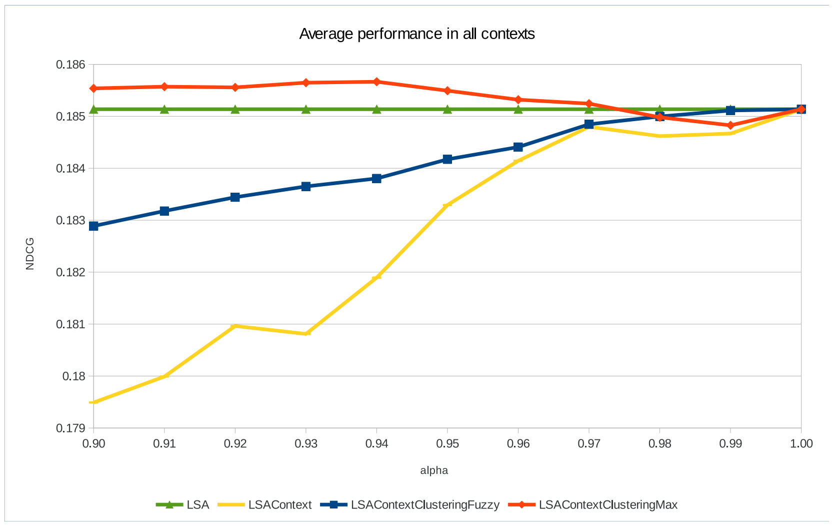

Figure 10: Average NDCG of the compared approaches in all contexts.

systems. In *Recommender Systems Handbook*, chapter 3, pages 217–253. Springer US, 2011.

[3] Gediminas Adomavicius and Alexander Tuzhilin. *Context-Aware Recommender Systems*, pages 191–226. Springer US, 2015.

[4] Charu C. Aggarwal. *Content-Based Recommender Systems*, pages 139–166. Springer International Publishing, 2016.

[5] Malak Al-Hassan, Haiyan Lu, and Jie Lu. A semantic enhanced hybrid recommendation approach: A case study of e-government tourism service recommendation system. *Decision Support Systems*, 72:97 – 109, 2015.

[6] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix factorization techniques for context aware recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys ’11, pages 301–304, New York, NY, USA, 2011. ACM.

[7] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. *A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment*, pages 299–331. Springer US, 2011.

[8] Ana Belen Barragans-Martínez, Marta Rey-Lopez, Enrique Costa-Montenegro, Fernando A. Mikic-Fonte, Juan C. Burguillo, and Ana Peleteiro. Exploiting Social Tagging in a Web 2.0 Recommender System. *IEEE INTERNET COMPUTING*, 14(6):23–30, NOV-DEC 2010.

[9] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[10] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modelling and User-Adapted Interaction*, 12(4):331–370, 2002.

[11] Jorge Castro, Rosa M. Rodríguez, and Manuel J. Barranco. Weighting of features in content-based filtering with entropy and dependence measures. *International Journal of Computational Intelligence Systems*, 7(1):80–89, 2014.

[12] Jorge Castro, Raciel Yera, and Luis Martínez. An empirical study of natural noise management in group recommendation systems. *Decision Support Systems*, 94:1 – 11, 2017.

[13] Sanket Chintapalli, Derek Dagit, Bobby Evans, Reza Farivar, Thomas Graves, Mark Holderbaugh, Zhuo Liu, Kyle Nusbaum, Kishorkumar Patil, Boyang Jerry Peng, et al. Benchmarking streaming computation engines: storm, flink and spark streaming. In *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International*, pages 1789–1792. IEEE, 2016.

[14] Luis Omar Colombo-Mendoza, Rafael Valencia-García, Alejandro Rodríguez-González, Giner Alor-Hernández, and José Javier Samper-Zapater. Recommetz: A context-aware knowledge-based mobile recommender system for movie showtimes. *Expert Systems with Applications*, 42(3):1202–1222, 2015.

[15] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. Semantics-aware content-based recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 119–159. Springer US, 2015.

[16] Toon De Pessemier, Cédric Courtois, Kris Vanhecke, Kristin Van Damme, Luc Martens, and Lieven De Marez. A user-centric evaluation of context-aware recommendations for a mobile news service. *Multimedia Tools and Applications*, 75(6):3323–3351, Mar 2016.

[17] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[18] Shuiguang Deng, Longtao Huang, and Guandong Xu. Social network-based service recommendation with trust enhancement. *Expert Systems with Applications*, 41(18):8075 – 8084, 2014.

[19] Ugo Erra, Sabrina Senatore, Fernando Minnella, and Giuseppe Caggianese. Approximate tf–idf based on topic extraction from massive message stream using the gpu. *Information Sciences*, 292:143 – 161, 2015.

[20] Alejandro Figueroa and Günter Neumann. Context-aware semantic classification of search queries for browsing community question–answering archives. *Knowledge-Based Systems*, 96:1 – 13, 2016.

[21] Asela Gunawardana and Guy Shani. *Evaluating recommender systems*, pages 265–308. Springer US, 2015.

[22] Alfred Hermida. Twittering the news. *Journalism Practice*, 4(3):297–308, 2010.

[23] Yehuda Koren and Robert Bell. *Advances in Collaborative Filtering*, pages 77–118. Springer US, Boston, MA, 2015.

[24] Nicolas Kuchmann-Beauger, Marie-Aude Aufaure, and Raphael Thollot. Context-aware question answering system, January 13 2015. US Patent 8,935,277.

[25] Pasquale Lops, Marco Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, chapter 3, pages 73–105. Springer US, 2011.

[26] Jie Lu, Qusai Shambour, Yisi Xu, Qing Lin, and Guangquan Zhang. A web-based personalized business partner recommendation system using fuzzy semantic techniques. *Computational Intelligence*, 29(1):37–69, 2013.

[27] Jesus Maillo, Sergio Ramírez, Isaac Triguero, and Francisco Herrera. knn-is: An iterative spark-based design of the k-nearest neighbors classifier for big data. *Knowledge-Based Systems*, 117:3 – 15, 2017. Volume, Variety and Velocity in Data Science.

[28] Judith Masthoff. Group recommender systems: Aggregation, satisfaction and group attributes. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 743–776. Springer US, 2015.

[29] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241, 2016.

[30] Cataldo Musto, Giovanni Semeraro, Pasquale Lops, Marco de Gemmis, and Georgios Lekkas. Personalized finance advisory through case-based recommender systems and diversification strategies. *Decision Support Systems*, 77:100 – 111, 2015.

[31] Lekha R Nair and Sujala D Shetty. Streaming twitter data analysis using spark for effective job search. *Journal of Theoretical and Applied Information Technology*, 80(2):349, 2015.

[32] T. T. S. Nguyen, H. Y. Lu, and J. Lu. Web-page recommendation based on web usage and domain knowledge. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2574–2587, 2014.

[33] Mehrbakhsh Nilashi, Othman Ibrahim, and Karamollah Bagherifard. A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. *Expert Systems with Applications*, 92:507 – 520, 2018.

[34] J.M. Noguera, M.J. Barranco, R.J. Segura, and L. Martínez. A mobile 3d-gis hybrid recommender system for tourism. *Information Sciences*, 215:37–52, 2012.

[35] Obaro Odiete, Tanvi Jain, Ifeoma Adaji, Julita Vassileva, and Ralph Deters. Recommending programming languages by identifying skill gaps using analysis of experts. a study of stack overflow. In *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, UMAP '17, pages 159–164, New York, NY, USA, 2017. ACM.

[36] Umberto Panniello, Alexander Tuzhilin, and Michele Gorgoglione. Comparing context-aware recommender systems in terms of accuracy and diversity. *User Modeling and User-Adapted Interaction*, 24(1-2):35–65, 02 2014.

[37] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 265–268, New York, NY, USA, 2009. ACM.

[38] Nish Parikh and Neel Sundaresan. Buzz-based recommender system. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 1231–1232, New York, NY, USA, 2009. ACM.

[39] L. Ponzanelli, G. Bavota, M. D. Penta, R. Oliveto, and M. Lanza. Prompter: A self-confident recommender system. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 577–580, Sept 2014.

[40] Luca Ponzanelli. Holistic recommender systems for software engineering. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 686–689, New York, NY, USA, 2014. ACM.

[41] Luca Ponzanelli, Simone Scalabrino, Gabriele Bavota, Andrea Mocci, Rocco Oliveto, Massimiliano Di Penta, and Michele Lanza. Supporting software developers with a holistic recommender system. In *Proceedings of the 39th Interna-*

tional Conference on Software Engineering, ICSE '17, pages 94–105, Piscataway, NJ, USA, 2017. IEEE Press.

[42] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[43] D. Rafailidis and A. Nanopoulos. Modeling users preference dynamics and side information in recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(6):782–792, 2016.

[44] Francesco Ricci. Contextualizing recommendations. In *ACM RecSys Workshop on Context-Aware Recommender Systems (CARS 2012). In: Conjunction with the 6th ACM Conference on Recommender Systems (RecSys 2012). ACM*, 2012.

[45] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[46] Bin Shao and Jiafei Yan. Recommending answerers for stack overflow with lda model. In *Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing*, ChineseCSCW '17, pages 80–86, New York, NY, USA, 2017. ACM.

[47] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Feature-weighted user model for recommender systems. In *Proceedings of the 11th international conference on User Modeling*, pages 97–106. Springer-Verlag, 2007.

[48] D. Wu, J. Lu, and G. Zhang. A fuzzy tree matching-based personalized e-learning recommender system. *IEEE Transactions on Fuzzy Systems*, 23(6):2412–2426, 2015.

[49] J. Xuan, X. Luo, G. Zhang, J. Lu, and Z. Xu. Uncertainty analysis for the keyword system of web events. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(6):829–842, 2016.

[50] Raciel Yera Toledo and Yailé Caballero Mota. An e-learning collaborative filtering approach to suggest problems to solve in programming online judges. *International Journal of Distance Education Technologies*, 12(2):51–65, 2014.

[51] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[52] X. L. Zheng, C. C. Chen, J. L. Hung, W. He, F. X. Hong, and Z. Lin. A hybrid trust-based recommender system for online communities of practice. *IEEE Transactions on Learning Technologies*, 8(4):345–356, Oct 2015.