Inf Retrieval J (2018) 21:369–409
https://doi.org/10.1007/s10791-018-9327-0

A language model-based framework for multi-publisher content-based recommender systems

Hamed Zamani[1] · **Azadeh Shakery**[1,2]

Received: 2 January 2017 / Accepted: 8 January 2018 / Published online: 6 February 2018

Abstract The rapid growth of the Web has increased the difficulty of finding the information that can address the users' information needs. A number of recommendation approaches have been developed to tackle this problem. The increase in the number of data providers has necessitated the development of *multi-publisher recommender systems*; systems that include more than one item/data provider. In such environments, preserving the privacy of both publishers and subscribers is a key and challenging point. In this paper, we propose a multi-publisher framework for recommender systems based on a client–server architecture, which preserves the privacy of both data providers and subscribers. We develop our framework as a content-based filtering system using the statistical language modeling framework. We also introduce AUTO, a simple yet effective threshold optimization algorithm, to find a dissemination threshold for making acceptance and rejection decisions for new published documents. We further propose a language model sketching technique to reduce the network traffic between servers and clients in the proposed framework. Extensive experiments using the TREC-9 Filtering Track and the CLEF 2008-09 INFILE Track collections indicate the effectiveness of the proposed models in both single- and multi-publisher settings.

Keywords Content-based recommender system · Adaptive filtering · Multi-publisher recommendation · Privacy preservation · Language models · Threshold optimization

Hamed Zamani is currently affiliated with the University of Massachusetts Amherst.

✉ Azadeh Shakery
shakery@ut.ac.ir

Hamed Zamani
h.zamani@ut.ac.ir

[1] School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran

[2] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

1 Introduction

The rapid growth of the World Wide Web increases the difficulty of finding the information related to users' information needs. Recommender systems or information filtering systems have been developed to tackle this problem, since 1990s. Information filtering systems notify people about new unstructured or semi-structured data (Belkin and Croft 1992).

Various filtering approaches, including content-based, collaborative, demographic, and hybrid, have been investigated in recommender systems (Bobadilla et al. 2013). Recently, as a result of increasing the amount of training data in recommender systems, collaborative filtering, which is mainly based on user-user similarities, has attracted much attention. However, the training data that can be used for training collaborative filtering systems is not always available, and thus collaborative filtering cannot be always used. On the other hand, content-based filtering systems only recommend items based on user-item similarities. Improving the performance of content-based filtering can also directly affect the performance of hybrid recommender systems. Existence of systems like Google Alert and the QSR system (i.e., query-specific web recommendation) Yang and Jeh (2006) indicate the importance of content-based information filtering systems which are the focus of this paper.[1]

The increase in the number of data providers, such as publishers and news agencies, makes it extremely difficult for users to keep track of their information needs from all data providers. The recent popularity of smart-phones and various mobile information filtering applications also motivates the study of multi-publisher recommender systems. Therefore, developing a multi-publisher recommendation framework could help both subscribers (users) and publishers (data providers) to achieve their goals: subscribers want to obtain all the information related to their needs and publishers want to send all relevant data to the subscribers. With more than one publisher in recommender systems (henceforth called *multi-publisher recommender systems*), we should make sure that a publisher does not have access to the documents of the other publishers. Furthermore, publishers should not be aware of the documents recommended by the other publishers, which are accepted by a subscriber. Therefore, preserving the privacy of both publishers and subscribers is a necessity in multi-publisher environments (see Sect. 3 for more details).

Lops et al. (2011) presented a high-level architecture for content-based filtering systems, which covers previous research studies in this area. Their proposed architecture is applicable to environments with only one data provider. Increasing the number of data publishers necessitate the development of a multi-publisher framework that preserves the privacy of both data providers and subscribers. For instance, publishers should not have access to the documents that were recommended to the users by other publishers. In this paper, we only study the privacy issues that are specific to multi-publisher environments. On the other hand, by growing the number of subscribers, scalability becomes an important issue in recommender systems. In this paper, we introduce a scalable framework for multi-publisher recommender systems based on a client–server architecture, which preserves the privacy of both publishers and subscribers.

In order to develop content-based filtering systems, various information retrieval approaches have been extensively studied. Although the language modeling framework (Ponte and Croft 1998) is among the state-of-the-art retrieval frameworks, not much

[1] In this paper, we focus on recommending textual data, such as news articles, scientific publications, etc.

attention has been paid to this framework in information filtering. Although a few studies (Bogers and van den Bosch 2007, 2009; Lavrenko et al. 2000; Zhang and Callan 2001a) have considered the language modeling framework for content-based filtering, we believe some components of content-based recommender systems (e.g., profile updating) are still needed to be deeply analyzed in the language modeling framework. In this paper, we use statistical language models to develop the proposed framework. In more detail, we employ a unigram language model for creating structured representations for documents and the KL-divergence retrieval model (Lafferty and Zhai 2001) for computing the similarity between documents and user profiles.

We further introduce a simple yet effective threshold optimization algorithm, called auto-adjust threshold optimization (AUTO), to calculate the dissemination threshold in the recommendation process. The similarity score of each new document and each profile is compared with this threshold to make an acceptance or rejection decision. Selecting an optimal threshold can have a significant effect on the filtering performance (Zhang and Callan 2001b). We then prove the stability of AUTO algorithm showing that it is not highly sensitive to the initial threshold value. Our experiments also confirm this theoretic finding.

As mentioned above, the proposed framework is designed based on a client–server architecture. Therefore, reducing the network traffic between the server-side and the client-side applications becomes important. To do so, only a language model sketch of each document is sent to the client-side applications. The language model sketch should be a good representation of the original document and the client-side application should be able to make the recommendation decision based on this sketch. To achieve this goal, we propose a language model sketching technique based on the language modeling framework.

To summarize, the main contributions of this paper include:

- Introducing and formalizing the problem of multi-publisher recommender systems
- Presenting a scalable privacy-preserving framework for multi-publisher recommender systems based on a client–server architecture
- Implementing the proposed framework using statistical language models for document filtering
- Designing a simple yet effective threshold optimization algorithm and proving its stability
- Proposing a language model sketching technique to reduce the network traffic between clients and servers

In our experiments, we use the OHSUMED collection (Hersh et al. 1994) which was used in the TREC-9 Filtering Track (Robertson and Hull 2000) and the INFILE collection which was used in the CLEF 2008 and 2009 Information Filtering Evaluation Track (Besançon et al. 2009). The experiments investigate the effectiveness of the proposed language modeling-based approach for adaptive filtering. Different aspects of the proposed adaptive filtering method (e.g., threshold optimization and profile updating) are extensively explored. We further demonstrate the effectiveness of the proposed framework in multi-publisher environments.

The rest of this paper is structured as follows: we review notable related work in Sect. 2. The multi-publisher recommender systems problem is introduced in Sect. 3. Section 4 describes the proposed multi-publisher framework for recommender systems. We further explain how we develop the proposed framework in Sect. 5. After that, we elaborate on the scalability of the proposed framework and prove the stability of the AUTO algorithm in Sect. 6. Section 7 reports the experimental results and investigates the effectiveness of the

proposed framework. We finally conclude our methodologies and results and discuss possible future directions in Sect. 8.

2 Related work

In this section, we first briefly mention the existing recommender system approaches, mostly those related to our research. We then review the work related to scalability and privacy-preservation in recommender systems. Finally, we provide a brief explanation for a number of mobile information filtering and document summarization methods.

2.1 Recommender systems

Recommender systems are partitioned into three main categories: demographic, collaborative, and content-based filtering (Bobadilla et al. 2013). Hybrid recommendation can also be counted as the fourth category, which refers to the combination of the mentioned types of recommender systems. In this subsection, we shortly review existing recommender systems with a particular focus on content-based recommender systems.

Demographic Filtering Demographic recommender systems use demographic features of users, such as age, gender, and nationality, for making recommendations. Krulwich (1997) created the first demographic recommender system in which demographic features were gathered directly from the users. Pazzani (1999) further proposed a framework which uses demographic information for recommendation. More recently, Beel et al. (2013) presented an overview of the data gathered from real-word recommender systems, which demonstrates the importance of considering demographic features for recommendation purposes.

Collaborative Filtering Collaborative filtering is the second type of recommender systems which has recently attracted much attention. The research studies in collaborative filtering are mostly based on machine learning and data mining approaches. Collaborative recommender systems focus on user-user and/or item-item relations for generating recommendation lists. To do so, many machine learning techniques, such as clustering, matrix factorization, and factorization machines, have been proposed. For instance, Wang et al. (2014) addressed the sparsity issue by clustering the users using the K-means algorithm. Hu et al. (2014) also proposed a clustering-based collaborative filtering method to handle big data. They first clustered the services (items) and then selected one of the clusters for making the recommendations. Matrix factorization techniques have been investigated as the most common technique in collaborative filtering (Koren and Bell 2011). The main goal of these techniques is capturing latent factors from user-item matrices. Hofmann (2004) proposed to employ probabilistic latent semantic analysis (pLSA) to learn latent factors for collaborative filtering. Singular value decomposition (SVD) is a popular matrix factorization approach for recommender systems (Takács et al. 2008).

A number of popular collaborative filtering approaches do recommendation based on neighborhood selection algorithms (Su and Khoshgoftaar 2009). The links between users in social networks have been also considered for neighborhood selection (Yang et al. 2014). A number of collaborative filtering approaches have also focused on information retrieval strategies, such as the vector space and probabilistic models (Soboroff and Nicholas 2000; Turney and Pantel 2010; Wang et al. 2010). More recently, Parapar et al. 2013 considered

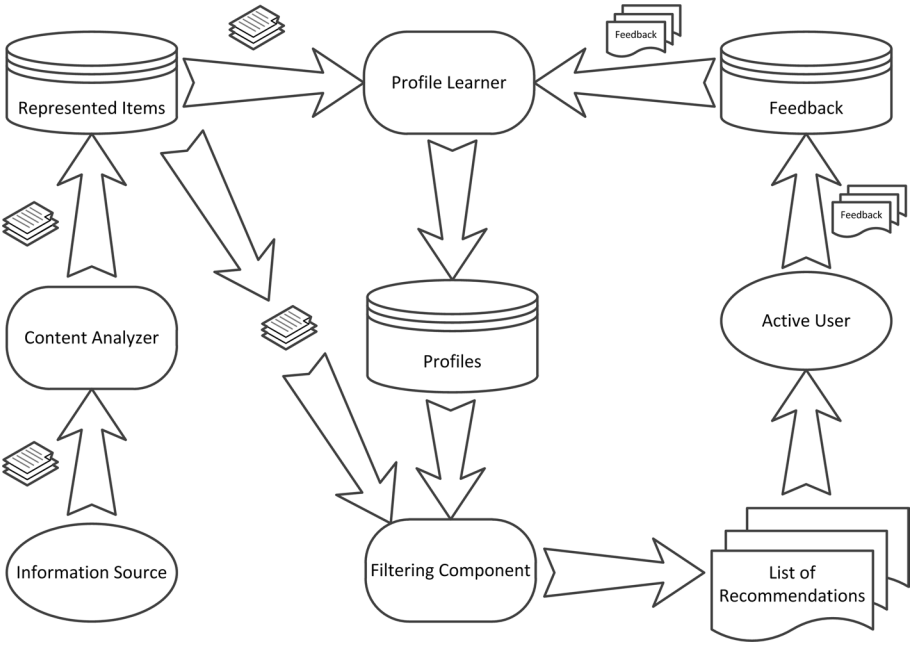

Fig. 1 The high-level architecture of content-based recommender systems introduced in Lops et al. (2011)

relevance-based language models (Lavrenko et al. 2001) for collaborative recommender systems.

Content-based Filtering The third type of recommender systems is content-based filtering which uses user-item similarities for recommendation purposes. In this kind of recommender systems, the recommended items to a given user u are the ones similar to those that were previously liked by the user u (Belkin and Croft 1992; Hanani et al. 2001; Lops et al. 2011). There are many research articles on content-based filtering in TREC Filtering Tracks from 1997 to 2002 (Hull 1997, 1998; Hull and Robertson 1999; Robertson and Hull 2000; Robertson and Soboro 2001; Robertson and Soboroff 2002). After that, a competition for evaluating mono-lingual and cross-lingual filtering systems was introduced in CLEF 2008 and 2009 Information Filtering Evaluation (INFILE) Track (Besançon et al. 2009). The Contextual Suggestion Track is another relevant shared task organized by TREC from 2012 to 2016 (Dean-Hall et al. 2012, 2013, 2014, 2015; Hashemi et al. 2016). In this task, each user profile consists of a set of past suggestions plus some contextual information. The task is to rank a number of suggestions for each user. Simultaneous to our work, CLEF 2017 introduces the NewsREEL shared-task (Lommatzsch et al. 2017) that focuses on providing news recommendation for multiple publishers. The framework used to evaluate this shared-task is fundamentally different from ours. The boldest difference is that there is a trustee system in the CLEF NewsREEL 2017 framework, named plista, that should be aware of the data from all publishers. However, in our framework, publishers directly send and receive data to and from users (clients) and such a trustee is not required. This makes our framework more applicable for real-world scenarios. It is noteworthy that this shared-task provides yet another evidence for the importance of recommendation in multi-publisher environments.

Lops et al. (2011) presented a high-level architecture for content-based filtering systems. Each content-based filtering system includes three main components: item representation, filtering component, and profile learner. This architecture is visualized in Fig. 1. Most of the previous methods use the vector space framework and the $tf - idf$ retrieval model for the item representation and the filtering components (Pazzani and Billsus 2007; van Metern and van Someren 2002; Mooney and Roy 2000; Lops et al. 2011). Castro et al. (2014) proposed an entropy-based approach to improve the simple $tf - idf$ method for content-based recommender systems. A few studies on language modeling have focused on content-based recommender systems. Bogers and van den Bosch (2007) compared different retrieval models in the content-based filtering task. They exploited the language modeling approach proposed by Ponte and Croft (1998) and showed that language modeling achieves a good performance in many cases. In Wang et al. (2010), relevance-based language modeling Lavrenko et al. (2001) is considered for forum-based news recommendation. In this paper, we also focus on the language modeling framework which allows us to have a strong statistical foundation to be able to extend our work in the future. In this paper, unlike in the existing ones, we analyze different aspects of content-based recommender systems, such as profile updating, in the language modeling framework.

As pointed out above, profile learner is another main component in content-based recommender systems. For updating user profiles, the Rocchio feedback algorithm (Rocchio 1971) has been extensively used in previous work (Allan 1996; Pazzani and Billsus 2007). Optimizing the dissemination threshold for making the acceptance or rejection decisions is one of the most influential parts in content-based filtering systems. KUN (Arampatzis et al. 2000) is one of the teams participated in the TREC-9 Filtering Track competition. They used a novel threshold optimization method which helps them to achieve the first rank in the competition. Zhang and Callan (2001b) also presented a threshold optimization algorithm based on maximum likelihood estimation (MLE) and achieved a better performance compared to KUN. Similar to KUN and MLE, there are a number of papers in the literature (Arampatzis et al. 2000; Arampatzis 2001; Arampatzis and van Hameran 2001; Zhang and Callan 2001b) with similar assumptions: they assumed that the distribution of relevant and non-relevant documents are normal and exponential, respectively. They further tried to estimate the parameters of these distributions. Arampatzis et al. (2009) used these assumptions to estimate where one should stop reading ranked lists in information retrieval.

Hybrid Filtering Hybrid filtering refers to a combination of more than one of the aforementioned types of recommender systems. Real-world recommender systems mostly use hybrid filtering because of their high accuracy (Bobadilla et al. 2013). Combination of content-based and collaborative filtering approaches is often used in hybrid systems (Barragáns-Martínez et al. 2010; Choi et al. 2012). Hybrid systems are usually based on probabilistic, machine learning, or bio-inspired methods. For instance, Linqi and Li (2008) used a genetic algorithm to combine the results of different recommender systems. Shinde and Kulkarni (2012) clustered users' opinions and selected one of the clusters with high-quality ratings to make recommendations. Chen et al. (2014) improved hybrid filtering systems for e-learning environments by using sequential pattern mining algorithms to filter the results of recommender systems. Since hybrid systems are combinations of individual filtering methods, improving any kind of the mentioned filtering categories may lead to improvements in the hybrid systems, as well.

According to Burke (2002), different types of recommendation approaches can be combined with different strategies. (1) Weighted hybrid recommender system combines different recommendation results obtained by different methods, in a weighted manner. (2) Mixed hybrid recommender system presents the results obtained by different recommendation types

to the users. (3) Feature combination-based hybrid system treats collaborative information as features and uses content-based approaches over this new feature space to make the final recommendations. (4) Cascade hybrid recommendation refines the recommendation list generated by a recommendation method using another recommendation approach. (5) Feature augmentation refers to a method that first categorizes or classifies the data and then uses different recommendation techniques to improve the performance. (6) Meta-level recommendation uses the model generated by one of the recommendation methods as an input to another recommendation approach.

2.2 Scalability in recommender systems

The exponential growth of data providers and subscribers in the Internet makes scalability an important issue in recommender systems. In Tryfonopoulos et al. (2009), a query indexing method was proposed for information filtering systems to tackle the scalability issues. The authors presented a data structure and an indexing algorithm for network-based filtering systems, such as peer-to-peer networks, where all peers can store data for making recommendations. Hence, it cannot be applied to all cases. In addition, they only evaluated their proposed method in terms of scalability. In Zimmer et al. (2008), a peer-to-peer architecture for information filtering systems was proposed. They introduced the MAPS architecture and showed the scalability of their framework. This architecture can only be applied to peer-to-peer environments. Tryfonopoulos and Andreescu (2011) modeled the cost of information in filtering systems for approximate information filtering. In approximate filtering systems, subscribers only monitor the selected data sources. Recently, distributed large-scale information filtering was studied in Tryfonopoulos et al. (2014). This methodology also only considers peer-to-peer networks.

2.3 Privacy-preservation in recommender systems

There are several privacy issues in various types of recommender systems that should be taken into account. Bonchi and Ferrari (2010) reviewed a number of privacy-preservation techniques for web recommender systems. According to their study, storing sensitive data (e.g., the information about users' preferences) in clients is a possible way to protect the privacy of users in content-based recommender systems. Erkin et al. (2012) proposed a privacy-preserving content-based filtering approach based on homomorphic encryption. In fact, their method obscures private information from the service provider. Parameswaran and Blough (2007) proposed to obfuscate sensitive data in such a way that it preserves individual privacy for collaborative recommender systems. Other techniques, such as randomized perturbation (Polat and Du 2003) and cryptography approaches (Zhan et al. 2010), have been also proposed for protecting the users' privacy in recommender systems. Cissée and Albayrak (2007) proposed a privacy-preserving method based on multi-agent systems. But, they did not evaluate their approach in terms of scalability and filtering performance. In this paper, we propose a client–server architecture which preserves the privacy of all participants in a multi-publisher filtering system. To the best of our knowledge, this is the first attempt to use this popular architecture for developing recommender systems.

2.4 Mobile information filtering

Mobile devices have recently become a primary platform for information access, retrieval, and filtering (Crestani et al. 2017). Mobile recommender systems are becoming more and more popular, especially when it comes to the area of travel and tourism (Braunhofer et al. 2014; Elahi et al. 2015; Gavalas et al. 2014; Ricci 2010). The availability of smart-phones with high-speed Internet access and position detection services (e.g., GPS) as well as the successful development of personal digital assistants (PDAs) have fostered the development of mobile information filtering systems (Ricci 2010). Mobile devices introduce some limitations, such as small screen sizes and relatively low processor power (Polatidis and Georgiadis 2014).

These unique characteristics as well as the availability of various similar or related smart-phone applications enhance the importance of recommending items from various data sources via mobile phones. For instance, several music streaming applications are available for smart-phones and recommending music from all of these sources could be interesting for many users. Such examples also provide a high potential for the applicability of multi-publisher recommender systems in mobile scenarios. We believe that studying multi-publisher recommender systems for mobile phones is a potential interesting future direction.

2.5 Summarization

Summarization is a research task in the natural language processing and information retrieval fields. Summarization in natural language processing (NLP) generally refers to generating a human-readable summary which describes the main content of a given document in brief (Jurafsky and Martin 2009; Saggion and Poibeau 2012). Summarization techniques in NLP can be partitioned into two categories: close and open summarizations. In close summarization, only the words and sentences from the original document can be used; however, open summarization can use additional resources. In close summarization, text categorization is a main technique for extracting informative sentences from long documents (Mani 1999; Neto et al. 2000). In Antiqueira and Oliveira (2009), a graph-based approach was also used to select high-ranked sentences. In open summarization, knowledge-based approaches have been widely explored. For instance, using lexical resources to generate summaries is counted as a knowledge-based approach for summarization (Saggion and Lapalme 2002; Li et al. 2010).

Summarization in information retrieval has been mostly used to reduce the index size in order to increase the scalability of retrieval systems, which was firstly introduced by Luhn (1958). After that, Brandow et al. (1995) showed that summarization for indexing may improve the precision, but it may lead to a dramatic drop in the recall value. Sakai and Spärck-Jones (2001) indicated that in addition to reducing the index size, summarization can be useful to improve pseudo-relevance feedback. The reason is that summarization can improve the precision value and pseudo-relevance feedback assumes that a small set of top-retrieved documents are relevant to the query, and thus improving the precision can significantly affect the feedback performance. They stated that the best compression ratio is 10–30% to achieve a good performance. To have a scalable retrieval, Müller et al. (2005) presented a summary-based retrieval in peer-to-peer networks. In Wan and Xiao (2010), a summarization technique was proposed based on the k-nearest neighbors algorithm. They

chose a small number of nearest documents to generate a better summary. All of the mentioned techniques and applications show the importance of summarization.

In this paper, we propose a sketching approach for language modeling framework (Ponte and Croft 1998) based on the KL-divergence retrieval model (Lafferty and Zhai 2001) that differs from existing summarization models.

3 Multi-publisher recommender systems: problem statement and motivation

Regarding the explosive growth of the Internet, the number of data providers has been tremendously increased in the World Wide Web. As a result, users would like to keep track of the subjects they are interested in, from different data sources. For instance, researchers in the field of computer science need to follow the papers published by ACM Digital Library, IEEE Computer Society, Elsevier Inc., etc. As another popular example, many users like to follow daily news published by different news agencies. Furthermore, different data providers can provide different items, but there might be common information that can be transferred among them to improve the recommendation performance. Previous work on cross-domain and multi-domain recommender systems and related tasks (Adomavicius and Tuzhilin 2005; Cantador et al. 2015; Cremonesi et al. 2011; Montazeralghaem et al. 2016b; Tang et al. 2012; Zamani et al. 2015; Zhang et al. 2010) can be considered as special cases of the recommendation task with more than one data provider. Therefore, following more than one data source by a user could potentially be an important problem. Increasing the number of data providers in the Web also enhances the importance of studying recommender systems in these environments. We refer to them as "*multi-publisher*" environments. To the best of our knowledge, this is the first attempt to study recommender systems in multi-publisher environments.

Sometimes the provided data by publishers is publicly available. In these situations, the problem of multi-publisher recommender systems can be cast to single-publisher or traditional recommender systems. In other words, the data published by different publishers can be put together as a source data and a recommender system can be employed for the recommendation process. Google Alerts[2] is a popular web service to make recommendations using public data in the Web. In fact, it is assumed that all data are publicly available and thus, privacy issues are not considered.

In some cases, data providers publish private information, the information that is not publicly available and users should register or pay to have access to the data. The existing recommender system frameworks assume to have full access to all the data and also the users profiles, which is usually constructed using the previously relevant recommended information. Having such an access is impossible in the real world recommender systems with private information since:

1. in many cases, the publishers are competing with each other and do not allow the others to access their databases.
2. user profiles usually include the previous recommended data which is provided by various publishers. Therefore, having full access to the user profiles is equivalent to having

[2] https://www.google.com/alerts.

access to parts of the data which might be private. Obviously, it is against the privacy right of publishers.
3. access of a publisher to the recommended data provided by other publishers is against the privacy right of users, since users may prefer to hide the information they received from other publishers.

To tackle the privacy issues in multi-publisher environments, a possible solution would be to deploy separate recommender systems for each data publisher. In other words, this solution casts a multi-publisher recommender system to n single-publisher recommender systems, where n denotes the total number of data publishers. Although this solution does not suffer from the three aforementioned problems, it has main drawbacks, such as: (i) this solution is not effective since the recommender system of each publisher does not use the information that users liked, but is recommended by the other publishers. (ii) developing a recommender system for each publisher is highly expensive and many data publishers cannot afford it.

To summarize, according to various potential usages, multi-publisher recommender systems should be considered as an important research problem. In this problem, efficiency, scalability, and privacy-preservation on both user and publisher sides are the key issues which should be taken into consideration.

4 Multi-publisher framework for recommender systems

In this section, we propose a multi-publisher framework based on client–server architecture in which publishers and users play the roles of servers and clients, respectively. Unlike the aforementioned recommendation frameworks, in this framework each user has two profiles: a public and a private profile. The purpose for using two profiles is to consider the data recommended by other publishers to improve the recommendation performance while preserving the privacy of both users and publishers. The public profile is the one that is explicitly created by users. This profile is publicly available for all the publishers. It can be easily constructed by the initial query issued by users during the registration in the system. However, the private profile is automatically created using the recommendation history and the user's feedbacks. This profile is not accessible by the publishers or other users. Therefore, public and private profiles are stored in the server-side (publisher) and the client-side (user) applications, respectively.

The high-level architecture of the proposed framework is presented in Fig. 2. According to the proposed framework,[3] each new item is stored and indexed in the server-side application. Similarity between the public profiles and a new generated item is also computed in the server-side application. After that, the server-side application makes the initial recommendation decisions and sends a sketch of the new generated item (a good and also a short enough representation of the item) to the client-side applications. As mentioned above, this application is stored and run in the users side and is responsible for making the personalized decisions. Hence, by computing the similarity between the private profile and the received language model sketch, the client-side application

[3] We modify the high-level architecture of single-publisher recommender systems proposed in Lops et al. (2011).

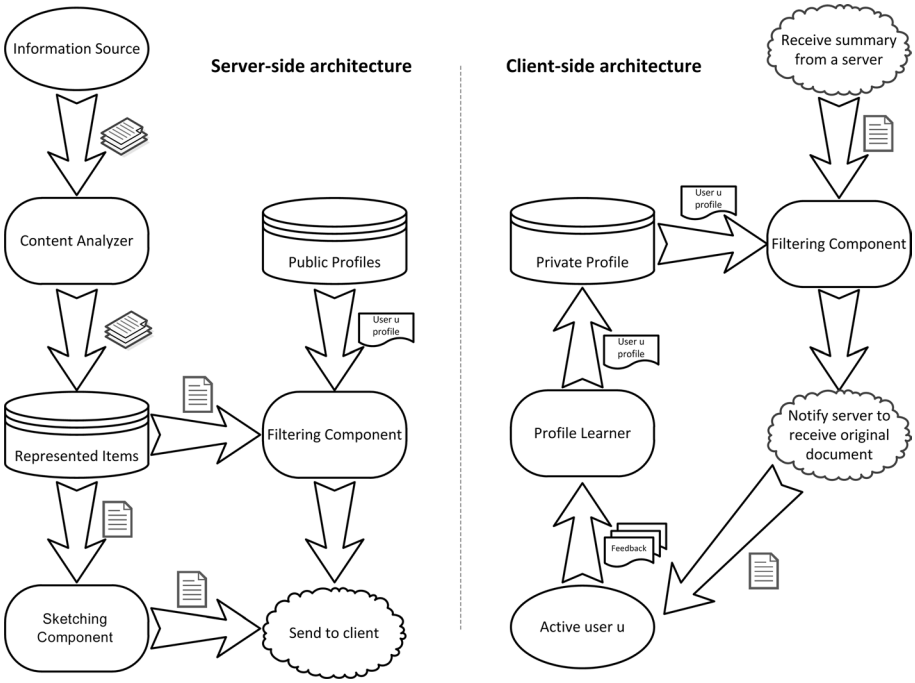

Fig. 2 High-level client–server architecture of recommender systems for multi-publisher environments

makes the recommendation decision locally and notifies the publisher or the server-side application. Based on the client-side decisions, the publisher will or will not send the original item to the user. It should be noted that updating the private profile based on the previous recommendations and the user's feedback is done in the client-side application.

In summary, as presented in Fig. 2, there are three main components in the server-side application:

Content Analyzer this component is responsible for performing pre-processing steps on the raw data (especially on textual data) for the next steps. In other words, Content Analyzer produces structured representations from unstructured data and stores them in the "Represented Items" database.
Filtering Component this component makes the initial recommendation decisions by calculating the similarity between the public profiles and the structured representation of new generated data. An initial recommendation will be made, if the computed similarity value is higher than or equal to a dissemination threshold parameter.
Sketching Component this component generates an item sketch which is a good representation of the original item. Note that the generated sketches do not need to be human-readable and their goal is to reduce the traffic load between server- and client-side applications.

There are also two primary components in the client-side application:

Filtering Component this component is similar to the Filtering Component in the server-side application. It computes the similarity between the received item sketch and the private profile.
Profile Learner this component automatically updates the private profile using the recommended items and also the user's feedback. The dissemination threshold is also updated in the Profile Learner component.

It is notable that the proposed architecture is a general framework and distinct server-side (publisher-side) applications can be implemented by each of the publishers. Therefore, this framework does not limit the data providers to share the same recommendation algorithms in the server-side applications. However, the data providers all should agree to share the same multi-publisher framework. The client-side application is shared among all data providers and the protocol for sending data from servers to clients should be the same for all publishers. In real-world scenarios, the publishers should trust the implementation of the multi-publisher framework.

5 A language model-based implementation for multi-publisher content-based recommender system

In this section, we first present a background on the use of statistical language models for document filtering and explain how we employ language models for implementing our multi-publisher framework. We further introduce our profile learner component (including a threshold optimization approach) in Sect. 5.2. As described in Sect. 4, the profile learner component is on the client side and thus, the profile updating approach should be highly efficient and incremental (i.e., it should not need the previously recommended documents). Therefore, many existing approaches for threshold optimization and profile updating reviewed in Sect. 2.1 cannot be applied in the multi-publisher setting. We finally introduce our language model sketching technique that reduces the network traffic between servers and clients for multi-publisher recommender systems.

5.1 Background: statistical language modeling for document filtering

Statistical Language Modeling (SLM) is a state-of-the-art retrieval framework which has been extensively studied in the information retrieval (IR) literature (Ponte and Croft 1998; Lafferty and Zhai 2001; Song and Croft 1999; Zhai and Lafferty 2004). SLM is a probabilistic framework with a well-defined structure which enables researchers to apply it to various retrieval tasks. Although SLM is highly popular in the IR community, there are a few studies (Bogers and van den Bosch 2007, 2009; Lavrenko et al. 2000; Zhang and Callan 2001a) in the content-based recommender systems which are developed using the language modeling framework. In addition, most of the existing language model-based methods are based on the query likelihood retrieval model proposed by Ponte and Croft (1998), which ignore profile updating which is a main component in content-based filtering. Therefore, we believe that the effectiveness of content-based recommender systems, including threshold and profile updating, based on the language modeling framework still needs to be studied.

In this subsection, we explain how the content analyzer and filtering components are implemented using the SLM framework.

5.1.1 Content analyzer

As pointed out in Sect. 4, the Content Analyzer component creates a structured representation of the documents. We employ unigram language models for modeling the documents. Although the unigram language model is very simple and is not very popular in natural language processing (NLP) applications, most of the language modeling-based methods in the information retrieval tasks are just using the unigram language model (Manning et al. 2008; Ponte and Croft 1998; Zhai and Lafferty 2004; Zhai 2008). In unigram language models, it is assumed that each word in a document is generated independently from the other ones.

The nature of content-based recommender systems is very similar to the document retrieval tasks, since the goal is to find similar documents to a given profile (like a query in IR) and recommend those documents with enough content similarity. The first reason to ignore more complex language models (e.g., bigram, trigram, etc.) in this paper is related to the problem of data sparseness. In other words, it is shown that the estimated complex language models is relatively inaccurate for retrieval processes (Manning et al. 2008; Zhai 2008). Another reason could be that document retrieval and recommendation processes are much simpler than a number of NLP applications, such as statistical machine translation, which need complex language models to show promising performance (Zhai 2008).

5.1.2 Filtering component

In the Filtering Component, we need to compute the similarity between a user profile P and a document D. To this aim, we compute the divergence between the language model of profile P (θ_P) and the language model of document D (θ_D). Note that each language model is a probabilistic distribution and thus, we can use Kullback–Leibler divergence (KL-divergence) (Lafferty and Zhai 2001) to compute the divergence between the distributions θ_P and θ_D. Therefore, the similarity between P and D is calculated as:

$$\begin{aligned} score(P,D) = -KLD(\theta_P,\theta_D) &= \sum_{w\in P} score_w(P,D) \\ &= \sum_{w\in P} - p(w|\theta_P)\log\frac{p(w|\theta_P)}{p(w|\theta_D)} \end{aligned} \tag{1}$$

where $score_w(P,D)$ represents how much the word w is influential in computing the similarity between D and P. To address the problem of zero probabilities and to normalize the documents length, the probability of each word w in the language model θ_D can be calculated using the Dirichlet prior smoothing method (Zhai and Lafferty 2004) as follows:

$$p(w|\theta_D) = \frac{|D|}{|D|+\mu}p_{ML}(w|D) + \frac{\mu}{|D|+\mu}p(w|C) \tag{2}$$

where μ and C denote the Dirichlet prior smoothing parameter and the reference language model, respectively. The reference language model is a general language model learned from a large corpus that is a good representative for the whole collection. $p_{ML}(w|D)$ is the maximum likelihood estimation of occurrence probability of term w in document D, which is calculated as $c(w, D)$ / $|D|$ where $c(w, D)$ and $|D|$ represent the frequency of term w in document D and the length of document D, respectively. Note that the language model of

each public profile is estimated using the maximum likelihood estimation. Estimating the language model of private profiles will be explained in Sect. 5.2.

As mentioned in Sect. 4, there are two Filtering Components in our framework, one in the server-side and the other one in the client-side applications (see Fig. 2). In the server-side Filtering Component, for each user u, $score(P_u, D)$ is computed, where P_u is the public profile of user u. If this value is higher than a dissemination threshold, the language model sketch of this document will be sent to the corresponding client-side application. The client-side Filtering Component calculates the similarity between the received sketch S and the private profile P_u^* ($score(P_u^*, S)$). If this value is higher than the dissemination threshold, the client-side program will accept the document based on its language model sketch and notify the publisher to send the original document.

5.2 Profile learner

Profile Learner is a client-side component which is responsible for updating the private profile and the dissemination threshold value. Client-side program has to update the user's private profile, according to the recommended documents and his/her feedback. Moreover, the dissemination threshold for accepting new documents may need to be readjusted. In the following subsection, we first present our profile updating strategy and then explain our simple threshold optimization algorithm.

5.2.1 Private profile updating

Unlike in web search, it is shown that the users will give explicit feedback in recommender systems. The most common feedback is binary: relevant or non-relevant. Similar to previous work in relevance feedback for ad-hoc information retrieval (Lavrenko et al. 2001; Lv and Zhai 2009; Montazeralghaem et al. 2016a; Zamani et al. 2016; Zhai and Lafferty 2001), we ignore non-relevant documents in profile updating. However, as discovered in Zagheli et al. (2017), non-relevant documents can be also used as negative feedback for further improvements. Exploiting semantic similarities for profile updating can be also used for improving the performance (Rahmatizadeh Zagheli et al. 2017; Zamani and Croft 2016, 2017).

In most of the relevance feedback methods, the feedback language model is linearly interpolated with the original query language model estimated by the maximum likelihood estimation. The feedback coefficient is set to a constant number in [0, 1] interval. In adaptive filtering, since the relevant documents are detected separately, selecting a constant feedback coefficient could not be a good choice. Intuitively, the weight of original query language model when there is only one feedback document available should be much higher than when many relevant documents are detected. To this aim, we propose to use a dynamic weighting instead of using a constant value as the feedback coefficient. We linearly interpolate the previous language model of private profile, the original profile (i.e., public profile), and the new feedback language model estimated using a new relevant document. Since no more information about the relevant documents is available, we can assume that all the feedback documents have the same weight. Considering this assumption, we can use the following equation to update the language model of the private profile.

$$\theta_{P_u^*}^{(n)} = \frac{n-1}{n}\,\theta_{P_u^*}^{(n-1)} + \frac{\alpha}{n}\,\theta_{P_u} + \frac{1-\alpha}{n}\,\theta_F \tag{3}$$

where θ_{P_u}, θ_F, and $\theta_{P_u^*}^{(n)}$ respectively denote the language model of public profile, the feedback language model, and the language model of private profile of user u when the nth relevant document is recommended. The free parameter α should be in the [0, 1] interval. This parameter controls the influence of initial profile in the language model of private profile. By the following theorem, we prove that in Eq. (3), all feedback documents are equally weighted.

Theorem 1 *All feedback documents are weighted equally by the incremental feedback algorithm presented in Eq. (3). This weight is equal to* $\frac{1-\alpha}{n}$*, where n denotes the number of feedback documents* ($n > 0$)*. The weight of the initial profile is always equal to* α*.*

Proof We prove the theorem by induction. For $n = 1$ (the base case), the private profile language model is computed as follows:

$$\theta_{P_u^*}^{(1)} = \alpha\theta_{P_u} + (1-\alpha)\theta_F \tag{4}$$

in which Theorem 1 clearly holds.

Let $k \in \mathbb{N}$ be given and suppose that Theorem 1 holds for $n = k$. Therefore, given the induction hypothesis, we can rewrite the private profile language model as follows:

$$\theta_{P_u^*}^{(k)} = \alpha\theta_{P_u} + \sum_{i=1}^{k} \frac{1-\alpha}{k}\theta_{F_i} \tag{5}$$

where θ_{F_i} denotes the feedback language model for the i^{th} feedback document.

Then, from Eq. 3 for $n = k + 1$ we have:

$$\theta_{P_u^*}^{(k+1)} = \frac{k}{k+1}\,\theta_{P_u^*}^{(k)} + \frac{\alpha}{k+1}\,\theta_{P_u} + \frac{1-\alpha}{k+1}\,\theta_F \tag{6}$$

Given the induction hypothesis in Eq. (5), we can rewrite Eq. (6) as follows:

$$\begin{aligned}
\theta_{P_u^*}^{(k+1)} &= \frac{k}{k+1}\left(\alpha\theta_{P_u} + \sum_{i=1}^{k}\frac{1-\alpha}{k}\theta_{F_i}\right) + \frac{\alpha}{k+1}\,\theta_{P_u} + \frac{1-\alpha}{k+1}\,\theta_F \\
&= \left(\frac{k\alpha}{k+1} + \frac{\alpha}{k+1}\right)\theta_{P_u} + \frac{k}{k+1}\sum_{i=1}^{k}\frac{1-\alpha}{k}\theta_{F_i} + \frac{1-\alpha}{k+1}\,\theta_F \\
&= \alpha\theta_{P_u} + \sum_{i=1}^{k}\frac{1-\alpha}{k+1}\theta_{F_i} + \frac{1-\alpha}{k+1}\,\theta_F
\end{aligned} \tag{7}$$

Therefore, Theorem 1 also holds for $n = k + 1$. By the principle of induction, it has been proven that the theorem holds for all $n \in \mathbb{N}$. □

Overall, we propose an incremental language model updating method. The feedback language model (θ_F) can be estimated using any of the relevance feedback methods. In this paper, we employ the mixture feedback model (Zhai and Lafferty 2001), a

state-of-the-art relevance feedback method, for private profile updating. In this method, it is assumed that words can be partitioned into two categories: topical words and background words. In mixture model, a hidden variable $Z_w \in \{0, 1\}$ is considered, which indicates whether word w is generated using the background model ($Z_w = 0$) or the topic model ($Z_w = 1$). Using the following expectation–maximization (EM) algorithm the language model θ_F can be estimated:

$$p^{(i)}(Z_w = 1) = \frac{(1-\lambda)\, p_\lambda^{(i)}(w|\theta_F)}{(1-\lambda)\, p_\lambda^{(i)}(w|\theta_F) + \lambda\, p(w|C)} \tag{8}$$

$$p_\lambda^{(i+1)}(w|\theta_F) = \frac{c(w;d)\, p^{(i)}(Z_w = 1)}{\sum_{w' \in d} c(w';d)\, p^{(i)}(Z_{w'} = 1)} \tag{9}$$

where the superscripts (e.g., i) show the iteration number in the EM algorithm. λ is a hyper-parameter controlling the probability of sampling words from the background language model C. $c(w; d)$ denotes the frequency of term w in the feedback document d. Note that this EM algorithm computes the feedback model for the last feedback document, and the introduced incremental feedback algorithm is used to combine it with the feedback models computed for previous feedback documents. Eq. (8) computes the expected value for the probability of a term w to be sampled from the topic model (i.e., $Z_w = 1$). Equation (9) maximizes the likelihood of sampling each term from the feedback language model.

5.2.2 AUTO: AUto-adjust threshold optimization algorithm

Most of the proposed threshold optimization algorithms for adaptive filtering (e.g., Zhai et al. 2000; Zhang and Callan 2001b) suffer from an essential drawback: they either use the scores of previous recommended documents in their computations or update the score of all previous documents after updating the user profile. Using previous scores is possible if the system does not update the users profiles, since after updating the user profiles all the previous scores become invalid. In addition, storing all the documents to compute all the scores after profile updating is very expensive and even impossible in many situations. This might be the reason that most of the previous works on threshold updating do not consider profile updating (Zhai et al. 2000; Zhang and Callan 2001b, 2003). However, profile updating is a key part in a content-based recommender system and it can significantly affect the performance (Maidel et al. 2008). Therefore, we propose a simple algorithm which solely uses the score of the last recommended document for updating the dissemination threshold.

Consider the case where the sketched representation of a new published document is sent to a client-side application. As shown in Fig. 2, $score(P_u^*, S)$ where P_u^* and S respectively denote the private profile and the received language model sketch, will be computed in the Filtering Component of the client-side application. If the score is higher than the threshold, it will be accepted for recommendation. As a result, accepting non-relevant documents could be a sign showing that the current threshold value is smaller than its optimal value. Increasing the threshold value in this situation logically will lead to improvement (or at least no change) in precision. On the other hand, rejecting a number of continuously received document sketches shows that the threshold is higher than its optimal value, and thus decreasing the threshold value will generally help to improve the recall.

Regarding the aforementioned arguments, we propose a general algorithm named AUtomatic Threshold Optimization (AUTO). In this paper, we propose two simple yet efficient and effective implementations for the AUTO algorithm: LAUTO and BAUTO.

In LAUTO, we try to implement linear search to find the optimum threshold value. To this aim, we consider two constant parameters c_1 and c_2. Using these two parameters, in each step the threshold value will be changed linearly. In other words, the function *Increment_Threshold* returns the sum of current threshold and parameter c_1; the function *Decrement_Threshold* returns the current threshold minus c_2. In contrast, BAUTO tries to perform binary search (Cormen et al. 2001) to find the optimum threshold value. In other words, it starts with a low threshold value and in each step, it exponentially increases the incremental value, i.e., *Increment_Threshold* returns the sum of current threshold and $\rho c_1^{(t)}$ where the superscript t denote the past iteration number, and ρ is a constant parameter. If the system does not recommend any document to the users for a while (see the second condition in "Algorithm 1"), the threshold value will be reduced by $c_1^{(t-1)}$ (i.e., $c_1^{(t+1)} = c_1^{(t)} - c_1^{(t-1)}$), and then the threshold value will be increased linearly by an increment of γ. It is similar to the binary search algorithm when the maximum value is not known. Considering the characteristics of linear and binary searches, we expect that BAUTO outperforms LAUTO.

A similar idea to BAUTO is also used in Slow Start and Fast Recovery methods in networking to adjust the window size for controlling the congestion and improving the network performance (Kurose and Ross 2009). Nodes in a network are not aware of the bandwidth, traffic, and congestion in their data path, and also they want to use the bandwidth as much as possible. Therefore, in TCP protocol, a similar method to BAUTO is used to adjust the window size of packets transmitted by each network node.

ALGORITHM 1: AUTO: AUto-adjust Threshold Optimization algorithm

Input: $Status$ of threshold optimization system, current threshold ($Threshold$), and the private profile of user ($UpdatedProfile$)
Output: The new threshold of system ($NewThreshold$)
if $Status == RECEIVING\_NONRELEVANT\_DOC$ **then**
    $NewThreshold = Increment\_Threshold(Threshold)$;
end
if $Status == NOT\_RECEIVING\_DOC$ **then**
    $NewThreshold = Decrement\_Threshold(Threshold)$;
else
    $NewThreshold = Threshold$;
end

5.3 Language model sketching

As shown in Fig. 2, Sketching Component is located in the server-side application. This component can be used for decreasing the amount of transferred data. The most challenging problems in this part are (i) how to generate document sketches, and (ii) what the size of each document sketch should be?

Document sketches are used in the Filtering Component of the client-side application. Therefore, good document sketches are those that their similarities to the private profile are very similar to the similarities between their original documents and the private profile. Note that our generated document sketches do not need to be human readable as they will

be processed by the client-side application. To sum it up, each generated document sketch should be a good representation for its original document, in terms of their similarity with the private profile. To do so, we define an *error* value which computes the difference of similarity between the sketch S generated from document D and the private profile P_u^* and similarity between D and P_u^*. Hence, the *error* value is defined as follows:

$$error\left(D, S, P_u^*\right) = \sum_{w_i \in P_u^*} error_{w_i}\left(D, S, P_u^*\right) \tag{10}$$

where $error_{w_i}$ shows how much error is produced by the word w_i. It is computed as:

$$error_{w_i}\left(D, S, P_u^*\right) = \left| score_{w_i}\left(D, P_u^*\right) - score_{w_i}\left(S, P_u^*\right) \right| \tag{11}$$

As pointed out in Sect. 5.1.2, the similarity between a document (or its sketch) and a profile is calculated using KL-divergence. According to the KL-divergence formula, low probability words have a small effect on the similarity output. As a result, we can omit the low probability words to generate a sketch for a document. In other words, we only keep the top m words with the highest probabilities in the document sketch. We assume that the document length ($|D|$), the Dirichlet prior parameter (μ), and the collection language model (C) are exactly the same as each other in both client-side and server-side applications. These assumptions make sense since the value of $|D|$ and μ can be mentioned in the generated sketch and also C usually does not change over time and thus, it can be imported in both applications. Therefore, each language model sketch generated by our technique, contains the top m high probability words with their own probabilities. We use these assumptions in the following calculations.

To answer the second question raised in the beginning of this subsection (i.e., how to set the size of language model sketches), we first compute the defined *error* value. According to Eq. (10), we need to compute $error_{w_i}$ as follows:

$$\begin{aligned} error_{w_i}\left(D, S, P_u^*\right) &= \left| score_{w_i}\left(D, P_u^*\right) - score_{w_i}\left(S, P_u^*\right) \right| \\ &= \left| -p\left(w_i|\theta_{P_u^*}\right) \log \frac{p\left(w_i|\theta_{P_u^*}\right)}{p(w_i|\theta_D)} + p\left(w_i|\theta_{P_u^*}\right) \log \frac{p\left(w_i|\theta_{P_u^*}\right)}{p(w_i|\theta_S)} \right| \\ &= \left| p\left(w_i|\theta_{P_u^*}\right) \log p(w_i|\theta_D) - p\left(w_i|\theta_{P_u^*}\right) \log p(w_i|\theta_S) \right| \end{aligned} \tag{12}$$

where $\theta_{P_u^*}$, θ_D, and θ_S denote the language model of the private profile, document language model, and its sketched language model, respectively. The first step in this equation comes from definition of the KL-divergence formula which is mentioned in Eq. (1).

There are three possibilities for each word w_i in the private profile:

$$\begin{cases} w_i \in D, w_i \in S \\ w_i \notin D, w_i \notin S \\ w_i \in D, w_i \notin S \end{cases} \tag{13}$$

In the first condition, w_i is observed in both document and the generated sketch. As mentioned above, the exact probability of each term will be written in the sketch. Therefore, according to Eq. (12) the value of $error_{w_i}$ for the first condition is equal to zero.

The second condition in Eq. (13) means that the word w_i is appeared neither in the document nor in the sketch. Therefore, the frequency of word w_i is equal to zero for both document and sketch. Since we assume that the background collection is the same in both server-side and client-side applications, the probability of word w_i in document language model and its sketched language model are equal and according to Eq. (12), the value of $error_{w_i}$ in the second condition will also be equal to 0.

In the last condition of Eq. (13), word w_i exists in the document, but is omitted during the sketching process. According to the calculations in Eq. (12) we have:

$$\begin{aligned} error_{w_i}\left(D, S, P_u^*\right) &= \left| p\left(w_i|\theta_{P_u^*}\right) \log p(w_i|\theta_D) - p\left(w_i|\theta_{P_u^*}\right) \log p(w_i|\theta_S) \right| \\ &= \left| p\left(w_i|\theta_{P_u^*}\right) \log \frac{p(w_i|\theta_D)}{p(w_i|\theta_S)} \right| \\ &= \left| p\left(w_i|\theta_{P_u^*}\right) \log \frac{\frac{|D|}{|D|+\mu} p_{ML}(w_i|D) + \frac{\mu}{|D|+\mu} p(w_i|C)}{\frac{|D|}{|D|+\mu} p_{ML}(w_i|S) + \frac{\mu}{|D|+\mu} p(w_i|C)} \right| \\ &= \left| p\left(w_i|\theta_{P_u^*}\right) \log \frac{\frac{|D|}{|D|+\mu} p_{ML}(w_i|D) + \frac{\mu}{|D|+\mu} p(w_i|C)}{\frac{\mu}{|D|+\mu} p(w_i|C)} \right| \\ &= \left| p\left(w_i|\theta_{P_u^*}\right) \log \frac{|D| p_{ML}(w_i|D) + \mu p(w_i|C)}{\mu p(w_i|C)} \right| \end{aligned} \tag{14}$$

In the above equations, Dirichlet prior smoothing is used (see Eq. (2)). In the third step, the maximum likelihood probability of word w_i in language model sketch S equals to zero so it is dropped.

According to the aforementioned sketching technique, we remove some of the words with low probability during sketching. In the following equations, without loss of generality, we assume that words are sorted in descending order in terms of their probability in θ_D. Since we only consider the top m words with highest probabilities in the language model sketch, the *error* value can be computed as follows:

$$\begin{aligned} error\left(D, S, P_u^*\right) &= \sum_{i=m+1, w_i \in P_u^*}^{d} \left| p\left(w_i|\theta_{P_u^*}\right) \log \frac{d p_{ML}(w_i|D) + \mu p(w_i|\theta_C)}{\mu p(w_i|\theta_C)} \right| \\ &\leq \sum_{i=m+1}^{d} \left| p\left(w_i|\theta_{P_u^*}\right) \log \frac{d p_{ML}(w_i|D) + \mu p(w_i|\theta_C)}{\mu p(w_i|\theta_C)} \right| \end{aligned} \tag{15}$$

In the first step of Eq. (15), we only compute the value of error for the words which are appeared in document and omitted in the sketch. In the second step, we omit the condition of $w_i \in P_u^*$ and thus, the result would be equal to or greater than the previous step. In other words, we compute an upper-bound for the *error* value.

Since the sketching process is done in the server-side application and the sketch size should be set on that side, $p(w_i|\theta_{P^*_u})$ is not available for computing the error value. Hence, we should estimate this value in the server-side application. $p(w_i|\theta_{P_u})$ where P_u is the public profile, is a possible estimation for this value. We leave analyzing other estimation possibilities, such as using the topic of the user profile, for future work.

Finally, the minimum value of m which satisfies the above inequality is selected in the server-side application and then, the language model sketch is generated.

6 Discussion

In this section, we first discuss the scalability of the proposed framework. After that, we present a proof for the stability of AUTO algorithm.

6.1 Scalability

Increasing in the number of subscribers in the World Wide Web makes scalability an important issue in recommender systems. In this subsection, we discuss the time and space complexity of the proposed framework and compare it with previous frameworks.

6.1.1 Time complexity

Assume that a data provider publishes N documents in each time unit. Also consider that the time complexity of Filtering Component and Profile Learner equal to $O(k_1)$ and $O(k_2)$, respectively. In the proposed framework, variable m in the sketching technique (i.e., the number of unique terms in language model sketches with positive probability) is discrete and less than the document length, so the server-side program can create all the possible sketches and there is no need to calculate m for each user. In other words, the sketching algorithm can be run off-line,[4] and thus it can be ignored in computing the time complexity. Suppose that on average each user profile will be retrieved for each document with the probability p. As a result the total server-side time complexity is given by:

$$O(NMpk_1)$$

where M is the total number of users. If we assume that among the retrieved documents on average each document sketch will be accepted in the client-side application with probability q, the total time complexity in client-side is equal to:

$$O(Np(k_1 + qk_2))$$

In the above equations, the worst case is where $p = q = 1$. This happens when all documents are detected as relevant documents.

In contrast, most of the previous methods (single-publisher recommender systems) use the architecture presented in Lops et al. (2011). In fact, they first compare each published document with all the user profiles and then if the system decides that the document is

[4] This means that for each new document, we can generate a limited number of language model sketches and for each user just use one of the generated sketches. In other words, there is no need to generate a language model sketch for each user.

relevant to a user, his/her profile should be updated. The total time complexity of server in this architecture will be equal to:

$$O(NMp(k_1 + k_2))$$

In almost all previous methods, k_2 is far greater than k_1 and because of the huge number of users in recommender systems, the difference between time complexity of our architecture and the architecture presented in Lops et al. (2011) is influential in the creation of a scalable recommender system. In addition, the user public profiles are updated rarely compared to their private profiles and many of the transactions on databases containing public profiles, only need the read access. In comparison, in (Lops et al. 2011) architecture, the system is updating the profiles repeatedly and in many cases, the concurrency problem will occur.

Furthermore, AUTO algorithm is significantly faster than most of the other threshold optimization algorithms. For instance, one of the state-of-the-art threshold optimization methods is based on maximum likelihood estimation (MLE) which is proposed in Zhang and Callan (2001b). The implementation of this algorithm is iterative and based on conjugate gradient descent method. The complexity of AUTO algorithm would be $O(1)$ for each retrieved document. It is worth noting that our experiments show the effectiveness of LAUTO and BAUTO algorithms in comparison with previous methods, such as MLE.

6.1.2 Space complexity

Most of the profile updating methods select a number of terms from retrieved relevant documents and add them to the user profiles (Pazzani and Billsus 2007). As a result, if there are M users in the system and each user on average receives K relevant documents, the space complexity in these methods would be $O(MK)$.

However, because the profile updating is in the client-side program, the space complexity of our proposed framework in server-side is $O(M)$ and in client-side is $O(K)$. In addition to the decrease in the amount of data which would be stored in server-side, this would also help the time complexity, since by increasing the size of databases, each transaction consumes more time.

6.2 Stability of AUTO algorithm

In this subsection, we prove that LAUTO algorithm becomes stable after a long time usage. This can be proven for BAUTO algorithm, similarly.

Consider that we use LAUTO algorithm in two different runs with two initial thresholds $T_1^{(0)}$ and $T_2^{(0)}$. Also, consider that $T_1^{(t)}$ and $T_2^{(t)}$ represent the threshold values of the first and the second runs in time t, respectively.

If after t time units, $T_1^{(t)}$ and $T_2^{(t)}$ become approximately equal, then after that the results for both of them would be similar. Definition 1 gives a formal definition for stability in threshold optimization algorithms.

Definition 1 *(Stability for threshold optimization algorithms)* A threshold optimization algorithm is stable if it starts with $T_1^{(0)}$ and $T_2^{(0)}$ as initial threshold values in two different runs and after t time units, $\left|T_1^{(t)} - T_2^{(t)}\right|$ becomes less than ε.

Lemma 1 *If $c_1, c_2 < \varepsilon$, then LAUTO algorithm would be stable.*

Proof Without loss of generality, we can assume that in time t:

$$T_1^{(t)} < T_2^{(t)} \tag{16}$$

We know that if $score(D, P_u) > T_1^{(t)}$, document D will be accepted by threshold $T_1^{(t)}$ and if $score(D, P_u) < T_1^{(t)}$, it will be rejected (We can replace document D with its sketch S without changing anything in the following calculations and conclusions). According to Inequality (16) and the mentioned facts, if a document is accepted by threshold $T_2^{(t)}$, it will be accepted by threshold $T_1^{(t)}$, too. Therefore, the number of retrieved non-relevant documents by threshold $T_1^{(t)}$ is greater than or equal to the number of accepted non-relevant documents by threshold $T_2^{(t)}$. Hence, according to the first condition of AUTO algorithm, the probability of increasing threshold of the first run in time $t+1$ is more than threshold of the second run, and since $c_1 < \varepsilon$, if $\left|T_1^{(t)} - T_2^{(t)}\right| \geq \varepsilon$, the value of $\left|T_1^{(t+1)} - T_2^{(t+1)}\right|$ will be smaller than $\left|T_1^{(t)} - T_2^{(t)}\right|$. Otherwise, if $\left|T_1^{(t)} - T_2^{(t)}\right| < \varepsilon$, after changing the thresholds $\left|T_1^{(t+1)} - T_2^{(t+1)}\right|$ would be less than or equal to ε (because $c_1 < \varepsilon$).

On the other hand, according to inequality (16), if a document is rejected by threshold $T_1^{(t)}$, it will be rejected by threshold $T_2^{(t)}$, too. Therefore, the number of rejected documents by threshold $T_2^{(t)}$ is more than the number of rejected documents by threshold $T_1^{(t)}$. As a result, the number of times which threshold of the second run is decreased by c_2 is more than the first run's. Since $c_2 < \varepsilon$, if $\left|T_1^{(t)} - T_2^{(t)}\right|$ is greater than or equals to ε, value of $\left|T_1^{(t+1)} - T_2^{(t+1)}\right|$ will be less than or equal to $\left|T_1^{(t)} - T_2^{(t)}\right|$. If $\left|T_1^{(t)} - T_2^{(t)}\right| < \varepsilon$ by changing the thresholds, the difference between them in time $t+1$ would stay less than ε and the stability status would not be changed.

Therefore, according to the aforementioned facts, the value of $\left|T_1^{(t)} - T_2^{(t)}\right|$ is decreasing and because always $\left|T_1^{(t)} - T_2^{(t)}\right| \geq 0$, T_1 and T_2 will eventually meet and thus, LAUTO algorithm is stable.

7 Experiments

In this section, we first introduce the datasets that are used in our experiments. We further explain the experimental setup and the evaluation metrics. We afterward report and discuss our results for single-publisher and multi-publisher recommender systems. At the end, we summarize our findings in these experiments.

7.1 Datasets

We use two standard test collections in our experiments: the OHSUMED collection[5] which was used in the TREC-9 Filtering Track and the INFILE[6] collection which was used in the

[5] Oregon Health and Science University MEDical collection.

[6] INformation FILtering Evaluation.

CLEF 2008 and 2009 INFILE Track. We also use the BBC News corpus (2002–2003) as a background collection in our experiments on the INFILE dataset.

7.1.1 OHSUMED dataset

The OHSUMED collection (Hersh et al. 1994) was collected from the United States National Library of Medicine's bibliographic database between 1987 and 1991. The collection which includes around 350,000 scientific articles is divided into two separate parts: training set (the articles published in 1987) and test set (the articles published from 1988 to 1991). We used the training set as the background collection in our experiments.

The OHSUMED collection was employed in TREC-9 Filtering Track (Robertson and Hull 2000) with two types of topics (queries): 63 OHSU topics and 4903 MeSH headings. In TREC-9 Filtering Track, 500 MeSH headings (called MSH) are selected to evaluate the participated methods. Each topic includes a title and a short description. We use both of them in our experiments. The average number of relevant documents for each OHSU and MSH topic are 51 and 249, respectively. The documents were judged by a number of physicians and medical librarians based on the outputs of interactive search (Hersh et al. 1994; Robertson and Hull 2000).

7.1.2 INFILE dataset

The INFILE collection (Besançon et al. 2008) includes news articles published in a 3-year period (2004–2006) by Agence France Presse (AFP). The collection includes around 1.5 million articles in three languages: Arabic, English, and French. Each document consists of many fields, such as content, headline, slug line, keywords, country, and city. We only use the news contents in our experiments.

For each of the mentioned languages, 100,000 documents were selected to evaluate mono-lingual and cross-lingual filtering systems in CLEF 2008 and 2009 INFILE Track (Besançon et al. 2009). The INFILE collection involves 50 topics for each language, covering two different categories: 30 topics focus on general news and events and the other 20 topics include scientific and technological issues. We considered the title and keywords of each topic in our experiments. Note that since we do not focus on cross-lingual information filtering in this research, only the English documents and topics are used in our experiments.

7.1.3 BBC news

There does not exist any training set in INFILE collection. Therefore, we use the English news articles published by BBC, which were crawled from the BBC News website[7] (Hashemi et al. 2010). To have a fair evaluation and comparison, we use around 20,000 documents published from Jan. 2002 to Dec. 2003, so that there is no conflict between the publication dates of BBC and INFILE documents. Note that we use this dataset as the background collection in the experiments on the INFILE dataset.

[7] http://bbc.co.uk/.

7.2 Experimental setup

In all the experiments, we use the Dirichlet prior smoothing method. The smoothing parameter μ is set to the average document length in each background collection. The parameters c_1 and c_2 in the LAUTO and BAUTO algorithms are set using 2-fold cross validation over the queries of each collection. We also follow the same procedure to set the parameters in the baseline methods. In all experiments except those explicitly mentioned, the size of users' profiles is set to 20 words. All documents were stemmed using the Porter stemmer. Stopwords were removed in all the experiments using the standard INQUERY stopword list. All the experiments were carried out using the Lemur toolkit.[8]

In the multi-publisher experiments, we split each of the collections to a number of sub-collections. Each of these sub-collections corresponds to one publisher. We consider two different strategies for splitting the collections: random splitting and similarity-based splitting using a three-pass K-Means algorithm as explained in Liu and Croft (2004). This algorithm is also implemented in the Lemur toolkit.

7.3 Evaluation metric

Several evaluation metrics have been proposed for evaluating filtering systems in TREC Filtering Tracks (e.g., T9P and T9U in TREC-9 and T10P and T10U in TREC-10) (Hull and Robertson 1999; Robertson and Hull 2000; Robertson and Soboroff 2002). F1-measure is the other metric that was also used in a number of TREC papers. CLEF INFILE Track (Besançon et al. 2009) introduced F1-measure as the main evaluation metric for filtering systems. F1-measure is equal to the harmonic average of precision and recall.

Similar to the CLEF INFILE Track, we consider F1-measure in our evaluations for two reasons: (i) F1-measure considers both precision and recall and is a popular metric in evaluation of the information filtering systems (Bobadilla et al. 2013; Ricci et al. 2011), and (ii) using a unified evaluation metric allows us to compare the results over different collections to have a complete analysis.

To evaluate multi-publisher recommender systems, in addition to using the total F1-measure, we use the mean and the standard deviation of the F1-measures achieved by various publishers. The F1-measure shows the recommendation performance from the users' perspective. Mean and standard deviation of F1-measures achieved by different publishers show the recommendation performance from the publishers' perspective.

Statistical significant differences of the results are determined using the two-tailed paired *t*test computed at a 95% confidence level with Bonferroni correction.

7.4 Experimental results and discussion

In the following subsections, we first evaluate the proposed methods when there is only one publisher available (i.e., the traditional adaptive filtering). Then, we study the effectiveness of the proposed method in multi-publisher environments.

[8] http://www.lemurproject.org/.

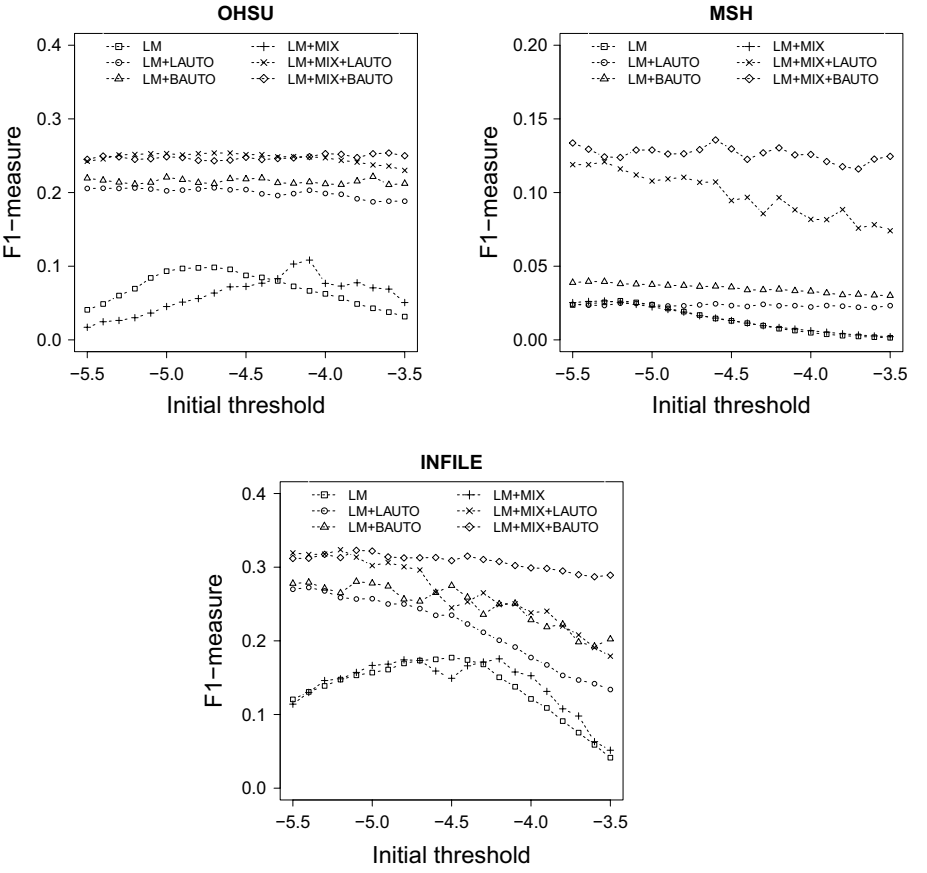

Fig. 3 Evaluation of the proposed language modeling approach with and without threshold optimization

7.4.1 Single-publisher environments

In this section, we first evaluate the effectiveness of the proposed threshold optimization algorithms and profile updating method on the aforementioned collections. Then, we compare the results achieved by the proposed method with those of the state-of-the-art adaptive filtering methods. In these experiments, since only one data provider is available, we assume that public and private profiles are the same and all the decisions are made in the server-side application (similar to the existing adaptive filtering methods). Therefore, we ignore document sketching in these experiments, since there is no need to send the sketch of documents to the client-side applications.

In the first set of experiments, we consider the following methods:

– *LM* this method only uses the language modeling framework based on the KL-divergence retrieval model without any profile updating or threshold optimization. In fact, this method only compares the similarity between the initial query of the user and each document with a constant dissemination threshold. The documents will be rec-

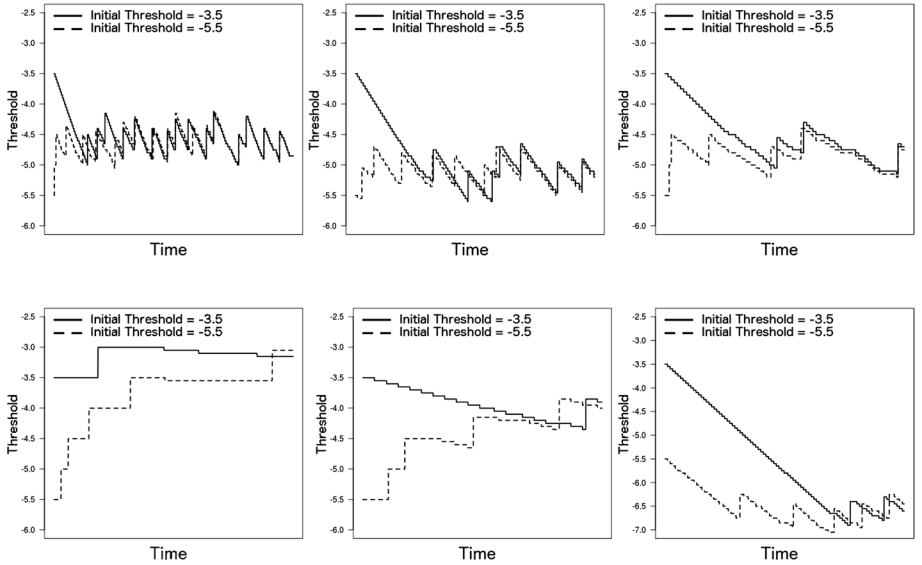

Fig. 4 Threshold changes using LAUTO for different initial threshold values in three random queries from the OHSU topics (top) and the INFILE dataset (bottom)

ommended to users, only if the computed similarities are above the constant threshold.

- *LM + LAUTO* similar to LM, this method also employs the language modeling framework with the same experimental setting. Unlike in LM, we adaptively modify the threshold value using the LAUTO algorithm introduced in Sect. 5.2.2.
- *LM + BAUTO* this method is exactly similar to LM + LAUTO, except for the threshold optimization algorithm; LM + BAUTO uses the BAUTO algorithm for updating the threshold values.
- *LM + MIX* this method uses the language modeling framework with a constant threshold value. In this method, user profiles will be updated after recommending each relevant document. Profile updating is done using the incremental mixture model (MIX) described in Sect. 5.2.1.
- *LM + MIX + LAUTO* this is a language modeling based adaptive filtering method which uses the LAUTO and MIX methods for threshold optimization and profile updating, respectively.
- *LM + MIX + BAUTO* this method is exactly similar to the previous one, except that it uses BAUTO algorithm for threshold optimization.

The F1-measure achieved by the mentioned methods with different initial dissemination threshold values are plotted in Fig. 3. According to this figure, LM is highly sensitive to the initial value of the threshold τ and it usually achieves the weakest performance compared to the other methods. In addition, the optimum results for the LM method in different collections do not occur in the same threshold value. In other words, the best initial threshold value for the LM method in the INFILE collection is not a good initial threshold value for MSH topics. Therefore, it is not possible to tune the initial threshold value on one collection and use it in another one. Thus, using a threshold optimization algorithm is necessary.

Table 1 The precision and the recall achieved by the proposed language model-based methods in different collections

Method	OHSU		MSH		INFILE	
	Prec.	Recall	Prec.	Recall	Prec.	Recall
LM	0.152	0.039	0.022	0.003	**0.444**	0.070
LM + LAUTO	0.176	0.227	0.117	0.012	0.224	0.146
LM + BAUTO	**0.189**	0.240	0.089	0.020	0.304	0.182
LM + MIX	0.076	0.077	0.021	0.003	0.357	0.097
LM + MIX + LAUTO	0.179	0.419	0.251	0.048	0.360	0.177
LM + MIX + BAUTO	0.180	**0.425**	**0.258**	**0.083**	0.334	**0.270**

The highest value in each column is marked in bold

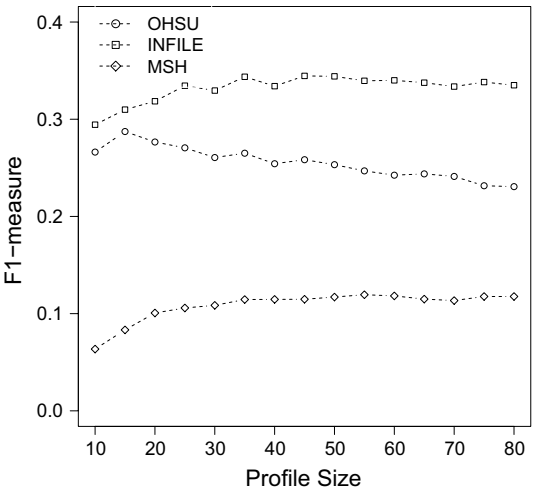

Fig. 5 The F1-measures achieved by LM + MIX + BAUTO with different profile sizes ($\tau = -4.0$)

Comparatively writing, LM + LAUTO and LM + BAUTO are more stable than LM in the OHSUMED collection (both OHSU and MSH topics) while the initial threshold value is varied. In contrast, LM + LAUTO is also very sensitive to the initial threshold value in the INFILE collection, but LM + BAUTO performs very stable in this collection. As mentioned in Sect. 5.2.2, LAUTO and BAUTO respectively develop the idea of the linear search and the binary search algorithms for finding the optimal threshold value, and thus BAUTO is expected to find the optimum threshold value faster than LAUTO. That is why BAUTO is also stable in smaller collections. Figure 4 shows the behaviour of the threshold value over time for three random queries in both OHSU and INFILE topics. According to this figure, two similar systems with very different initial threshold values meet each other in the OHSU topics, but they are far away from each other in the INFILE topics. The reason is related to the size of the collections. As theoretically proved in Sect. 6.2, the dissemination threshold values of different runs with various initial threshold values eventually meet each other. In other words, in the INFILE collection, more documents are needed to have stable results for different initial threshold values. To wrap it up, when the collection size is relatively large, the threshold values eventually meet each other which will lead to stable results.

Table 2 F1-measure achieved by different weighting methods in profile updating

Method	OHSU	MSH	INFILE
Constant weighting	0.2480	0.0924	0.2751
Dynamic weighting	**0.2527***	**0.1258***	**0.3088***

The superscript * denotes statistically significant differences between the results. The highest value in each column is marked in bold

Table 3 Comparing the proposed method with the baselines, in terms of F1-measure

Method	OHSU	MSH	INFILE
LM + BAUTO	**0.2114***	0.0326	**0.2276***
KUN	0.0656	0.0396	0.1169
MLE	0.0908	**0.0470***	0.1328
LM + MIX + BAUTO	**0.2528***	**0.1256***	**0.2986***
KUN + Profile Updating	0.1324	0.0951	0.1617
MLE + Profile Updating	0.1532	0.1041	0.1704

The superscript * indicates the results that are significantly higher than those achieved by the other methods. The highest value in each column is marked in bold

Based on Fig. 3, using the profile updating algorithm (MIX) without threshold updating might lead to decrease in the performance. Considering the KL-divergence formula for computing the similarity between profiles and documents (see Eq. (1)), it can be found out that by increasing the number of unique words in the profile, the output of the KL-divergence formula will dramatically change; since the summation is over the unique profile terms. This makes it necessary to update the threshold values when profiles are updated. As shown in Fig. 3, using the profile updating algorithm simultaneously with one of the threshold optimization algorithms improves the results, significantly. This shows the effectiveness of the proposed profile updating and threshold optimization algorithms for adaptive information filtering. In general, we can conclude that LM + MIX + BAUTO is the best and the most stable method compared to the other ones.

To complete the analysis, precision and recall achieved by each of the mentioned methods are reported in Table 1. In this set of experiments, the initial threshold value is set to − 4.0. According to Table 1, threshold optimization algorithms increase the recall value, significantly, since by having a high constant threshold value, the system may recommend a few number of documents. The recall value achieved by the BAUTO algorithm is always higher than those obtained by the LAUTO algorithm.

To investigate the influence of the profile size on the performance, Fig. 5 plots the F1-measure achieved by LM + MIX + BAUTO (i.e., the best performing method) where the initial threshold value is set to − 4.0.[9] According to Fig. 5, by increasing the profile size, the performance is generally increased in the INFILE and MSH topics and becomes stable when the profile size is greater than 40. In contrast, by increasing the profile size, the performance in OHSU topics is dropped when the profile size is higher than 15. As

[9] As shown in Fig. 3, LM + MIX + BAUTO is not highly sensitive to the initial threshold value, and thus fixing the threshold value will not significantly affect the filtering performance.

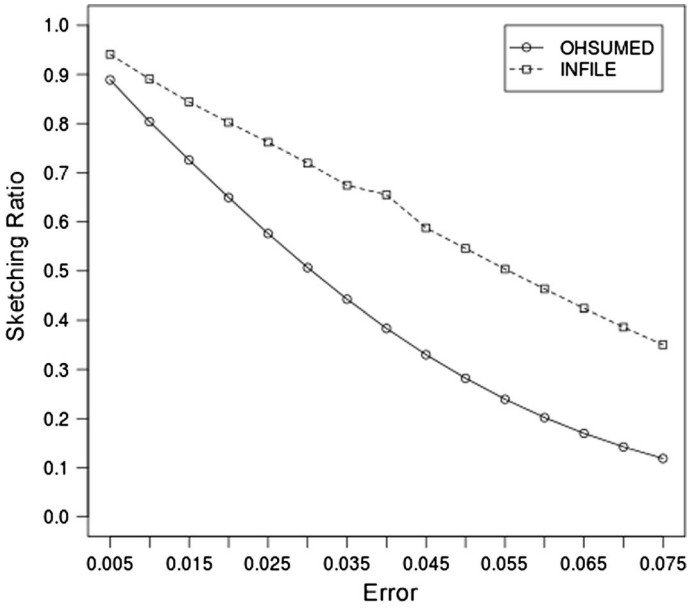

Fig. 6 Sketching ratio for different error value in OHSUMED and INFILE datasets

shown in Table 1, the precisions achieved by LM + MIX + BAUTO in the INFILE and MSH topics are higher than the recall values. But, in the OHSU topics, precision is much less than recall. On the other hand, increasing the profile size usually will lead to increase in the recall values, since more terms will be added to the profile. Regarding the definition of the F1-measure, this metric is biased toward the lower value. Therefore, by increasing the profile size the recall in all the collections will be increased and since the recall values in INFILE and MSH topics are lower than the precision values, increasing the profile size will increase the F1-measure, while it is not the case for the OHSU topics.

Most of the existing pseudo-relevance feedback algorithms combine the original query model with the feedback model using linear interpolation with a constant weight. Table 2 reports the best results achieved by constant weighting and the proposed incremental dynamic weighting (see Sect. 5.2.1). In this experiment, the initial threshold value and the profile size are set to − 4.0 and 20, respectively. According to this table, dynamic weighting significantly outperforms constant weighting in all the collections. The reason is that in constant weighting, weights of the new recommended documents are much higher that those of the previous ones; while in the proposed dynamic weighting, all relevant documents are weighted equally.

In the next set of experiments, we compare our best performing method (LM + MIX + BAUTO and LM + BAUTO for experiments with and without profile updating) with two baselines: (i) MLE (Zhang and Callan 2001b), a state-of-the-art adaptive filtering method which uses maximum likelihood estimation for threshold updating, and (ii) the best performing method in TREC-9 Filtering Track [i.e., KUN method Arampatzis et al. 2000]. KUN and MLE assume that scores of relevant and non-relevant documents are sampled from normal and exponential distributions, respectively. Their main difference is in estimating the parameters for these distributions. The CLEF 2009 INFILE Track was not as successful as the TREC Filtering Track. For the English topics, there was only one participant (UOWD)

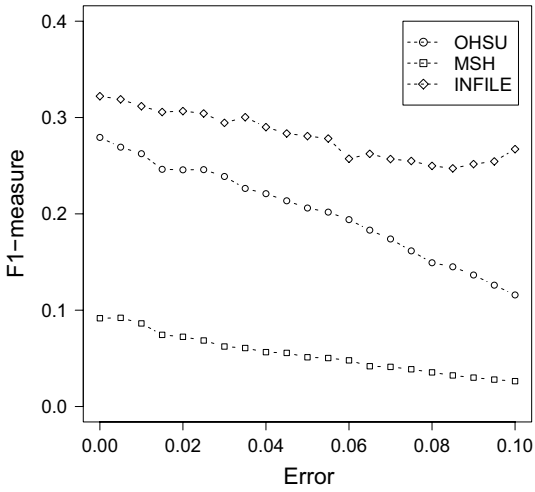

Fig. 7 F1-measure achieved by the proposed multi-publisher framework

in the adaptive filtering task, which did not perform well (UOWD achieved an F1-measure of 0.0100 in the adaptive filtering task). Therefore, we do not consider this method as the baseline.

Table 3 reports the results of the proposed method and the baselines in each collection with and without profile updating. As shown in this table, the proposed method without profile updating (LM + BAUTO) significantly outperforms the baselines in OHSU and INFILE datasets. Furthermore, our model with profile updating (LM + MIX + BAUTO) performs better than the baselines in all cases, significantly. All the baselines suffer from low recall and since F1-measure is biased toward the lower value of precision and recall, their performance are far below the results achieved by our proposed method. It should be noted that the KUN method that participated in TREC-9 Filtering Track employs several pre-processing steps and different technologies (such as query zoning). To have a fair comparison, we only consider the technologies that are also used in MLE and our proposed method. Therefore, the results of the complete KUN method can be higher than those presented in Table 3.

7.4.2 Multi-publisher environments

In this subsection, we first analyze the proposed sketching method in multi-publisher environments. We further compare the performance of the proposed multi-publisher recommender system with n single-publisher recommender systems, where n is the number of publishers. In all of these experiments, we employ the LM + MIX + BAUTO method which performs better than the other methods in single-publisher experiments. The initial threshold value and the profile size are set to -4.0 and 20, respectively.

Since each document sketch includes a number of terms with their probabilities, to show the behaviour of the proposed sketching technique, we define sketching ratio as below:

$$\textit{Sketching ratio} = \frac{\#\textit{ of unique words in the sketch}}{\#\textit{ of unique words in the original document}}$$

According to Fig. 6, by increasing the error value (see Eq. (15)) the sketching ratio will be decreased, which is expected. In other words, by increasing the error value, the sketching

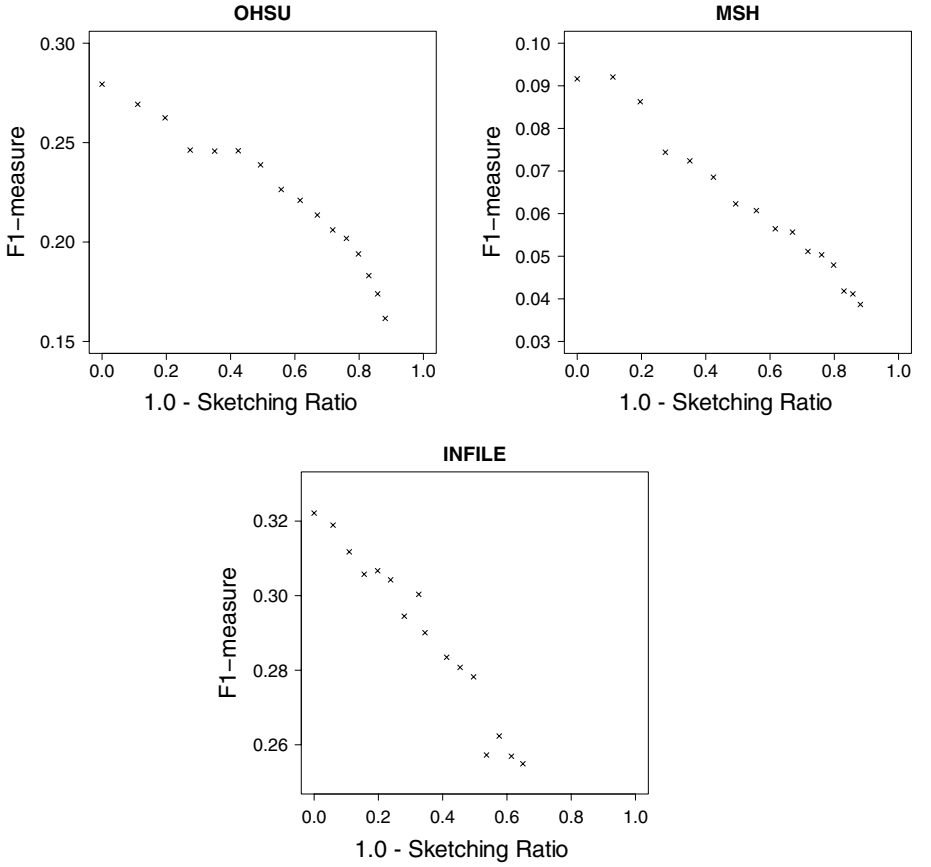

Fig. 8 Efficiency (i.e., "1.0 − *sketching ratio*" which should have a strong correlation with the network traffic) versus effectiveness (i.e., F1-measure) in the proposed multi-publisher framework

Table 4 The results achieved by the proposed methods in different collections, when each collection is randomly split into 10 sub-collections ($\tau = -4.0$)

	SP	MP-0.000	MP-0.030	MP-0.050	MP-0.075
OHSU					
F1	0.1860	0.2793	0.2388	0.2060	0.1615
μ_{F1}	0.1857	0.2792	0.2388	0.2046	0.1603
σ_{F1}	0.0002	0.0003	0.0002	0.0001	0.0004
MSH					
F1	0.0302	0.0916	0.0622	0.0511	0.0386
μ_{F1}	0.0302	0.0915	0.0621	0.0507	0.0384
σ_{F1}	0.0001	0.0001	0.0001	0.0001	0.0001
INFILE					
F1	0.2283	0.3221	0.2944	0.2807	0.2549
μ_{F1}	0.2267	0.3216	0.2941	0.2789	0.2548
σ_{F1}	0.0005	0.0004	0.0006	0.0005	0.0003

Table 5 The results achieved by the proposed methods in different collections, when each collection is split into 10 sub-collections using the K-means algorithm ($\tau = -4.0$)

	SP	MP-0.000	MP-0.030	MP-0.050	MP-0.075
OHSU					
F1	0.1928	0.2793	0.2388	0.206	0.1615
μ_{F1}	0.1921	0.2790	0.2386	0.2057	0.1595
σ_{F1}	0.0011	0.0004	0.0003	0.0001	0.0003
MSH					
F1	0.0369	0.0916	0.0622	0.0511	0.0386
μ_{F1}	0.0361	0.0916	0.0621	0.0501	0.0379
σ_{F1}	0.0014	0.0002	0.0003	0.0003	0.0003
INFILE					
F1	0.2357	0.3221	0.2944	0.2807	0.2549
μ_{F1}	0.2349	0.3218	0.2940	0.2784	0.2542
σ_{F1}	0.0016	0.0002	0.0007	0.0007	0.0003

technique will ignore more words, and thus the sketching ratio will be decreased. It is also crucial to know how the performance of the multi-publisher framework will change with varying the error value. Figure 7 demonstrates the performance of the proposed framework by increasing the error value in the sketching algorithm. As plotted in Fig. 7, by increasing the error value, the performance generally decreases, which is expected.

Since the sketching ratio should have a strong correlation with the network traffic (because the language model sketches are sent from the servers to the clients), we can consider it as an efficiency metric in the proposed framework.[10] The scatter plots in Fig. 8 visualize efficiency vs. effectiveness achieved by the proposed framework. According to the aforementioned reasons, we consider "$1.0 - \textit{sketching ratio}$" as an efficiency metric and F1-measure as an effectiveness metric. According to this figure, by increasing the efficiency of the system, the effectiveness metric generally decreases, which is expected. An interesting observation is that for example in the OHSU topics, when we double the efficiency metric (e.g., 0.4 to 0.8), the effectiveness metric drops from 0.25 to 0.20 (equivalent to 20% relative change). We can capture almost similar behaviours in other collections. Therefore, there is a trade-off between efficiency and effectiveness in our framework, and thus the parameters should be set based on various issues, such as user preferences and network capacity.

According to Fig. 6, when the sketching ratio is equal to 0.5 (which means that the sketch size is equal to half of the document size), the error value should be set to around 0.03 and 0.05 for OHSUMED and INFILE collections, respectively. When the threshold is set to 0.075 the size of sketches are around 10 and 33% of the documents size in OHSU and INFILE collections, respectively. To have a deep understanding of the results, we select these error values for the next set of experiments.

We randomly partition the documents of each collection into 10 subsets. Each subset corresponds to a publisher. A possible solution would be using an existing single-publisher

[10] It should be noted that there are several ways to define efficiency in such systems. In some sense, network traffic could be a measure for efficiency. The computational load on the server-side or even on the client-side applications could be also considered as a measure to evaluate efficiency. Since the computational cost in server-side and client-side applications are theoretically analyzed in Sect. 6.1, in this set of experiments, we focus on the network traffic as an efficiency metric.

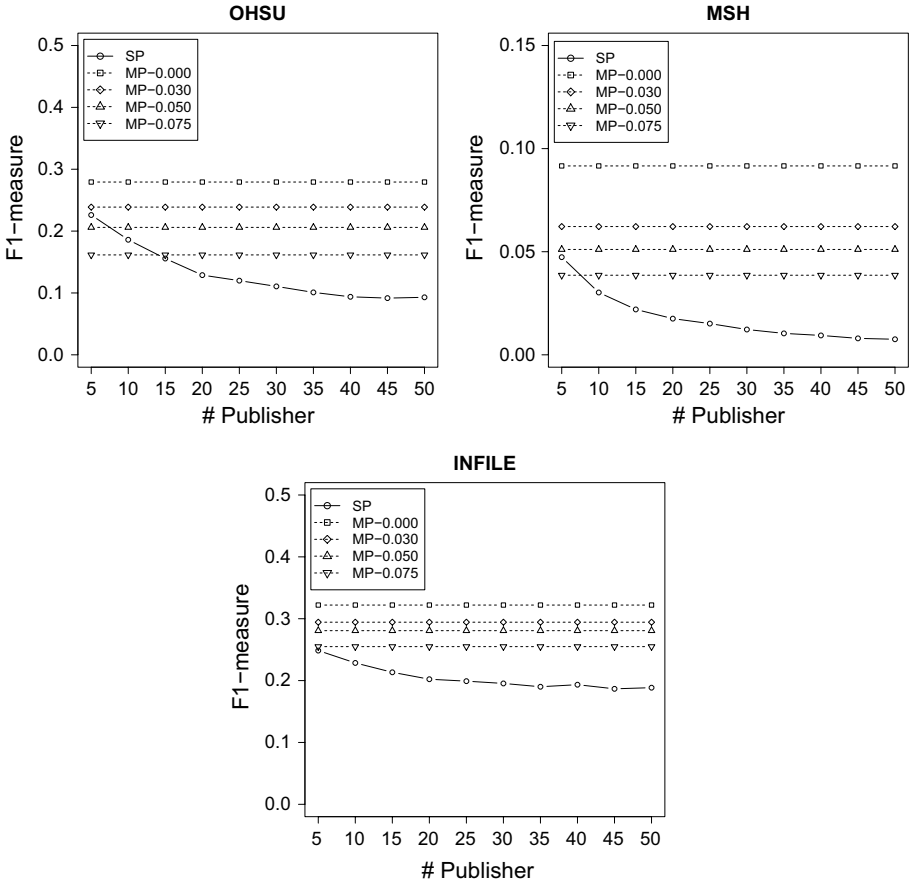

Fig. 9 Performance of the proposed multi-publisher framework by increasing the number of publishers

framework for each publisher. Table 4 reports the results of 10 single-publisher recommender systems (SP) and the proposed multi-publisher framework with different error values (MP-X where X denotes the value of the error parameter). In this table, F1 stands for the total F1-measure from the users' perspective. μ_{F1} and σ_{F1} respectively show the mean and standard deviation of the F1-measures achieved by the publishers. These values can indicate the satisfaction of publishers. According to Table 4, MP-0.000, MP-0.030, and MP-0.050 significantly outperform the single-publisher baseline. MP-0.075 also performs better than SP in INFILE and MSH topics. Note that according to Fig. 6, when the error value is equal to 0.075, the sketching ratio in OHSUMED collection is around 10%. Table 4 also demonstrates that μ_{F1} and F1 values are close to each other and the standard deviation of F1-measures achieved by the publishers are quite small. This shows that in this experiment the results achieved by all publishers are close, and thus the proposed method performs fair for all the publishers.

Random partitioning of documents can be considered as a simulation scenario when all data providers publish similar (in terms of generality and topics) documents. This scenario could be suitable for general news agencies. In contrast, in some cases, data providers

publish their own specific types of data. For instance, one data provider publishes the papers in the area of computer science, but the other one publishes electronic engineering publications. To also evaluate the proposed method for such a scenario, we split each collection into 10 subsets using the K-Means algorithm. We employed a three-pass K-means algorithm similar to what was previously done by Liu and Croft (2004) for cluster-based language model smoothing. In this experimental setup, the documents in each subset are more likely to be similar. The results are reported in Table 5. The F1-measures achieved by the MP-X methods is exactly similar to the results reported in the random splitting experiment (see Table 4), since the MP-X methods learn and store profiles in the client-side application, and thus the profiles are independent of how we split a collection for performing this experiment. One difference with the previous experiment is that the improvements achieved by the MP-X methods compared to those obtained by the SP method in the random splitting experiment are higher than those in this experiment. The reason is that similar documents are published by each publisher, and thus the SP method can also learn a good profile for each user. It should be noted that the standard deviations of the F1-measure achieved by SP are also increased in this experiment, compared to the random splitting experiment. The reason is that in this experimental setting, a number of publishers only have a few relevant documents for some topics, and thus, in this case SP method cannot learn good profiles for users.

Figure 9 shows the performance of SP and MP-X methods with different number of publishers. In this experiment, the documents are also randomly distributed among the publishers. According to Fig. 9, the performance of SP drops by increasing the number of publishers. However, the performance of MP-X method is stable since, the profile updating and recommendation is done in the client-side which is independent of the number of publishers.

7.5 Summary

In this subsection, we summarize the experimental findings in this paper. We evaluate our methods using two standard filtering collections: the OHSUMED collection that was used in TREC-9 Filtering Track and the INFILE collection that was used in CLEF 2008 and 2009 Information Filtering Evaluation (INFILE) Track.

We first evaluate the proposed method for single-publisher environments, i.e. where only one data provider is available. The first set of experiments investigate the importance of optimizing the dissemination threshold value and updating the user profiles in content-based filtering. It is shown that the best and the most stable results are achieved when we use the BAUTO algorithm for threshold optimization and the incremental mixture model for profile updating. We then study the effects of profile size on the filtering performance and show that profile size is a collection-dependent parameter and should be set appropriately according to the collection and the users' needs. We also show that the incremental feedback with dynamic weighting outperforms the same feedback method with constant weighting. We further compare the proposed method with state-of-the-art baselines and show that the proposed method significantly outperforms the baselines.

In the next set of experiments, we study the effectiveness of the proposed framework for multi-publisher environments. We first study the sensitivity of the sketching algorithm to the *error* parameter. We also investigate efficiency vs. effectiveness in the proposed framework. At the end, we demonstrate that the proposed multi-publisher framework outperforms individual single-publisher recommender systems, significantly.

8 Conclusions and future work

In this paper, we introduced multi-publisher recommender systems and developed a scalable privacy-preserving framework for multi-publisher environments based on a client–server architecture. We inherited the practical and theoretical advantages of statistical language modeling (SLM) to implement the proposed framework. We further introduced AUTO algorithm, a stable auto-adjust threshold optimization algorithm, for optimizing the dissemination threshold value in content-based recommender systems. For profile updating, we proposed an incremental feedback approach which can be used with any relevance feedback approach in the language modeling framework. We also proposed a sketching technique to reduce the network traffic between server-side and client-side applications.

We evaluated the proposed framework using two standard collections: the TREC-9 Filtering Track's collection and the CLEF 2008 and 2009 INFILE Track's collection. The experiments investigated the effectiveness of the proposed language modeling-based method for content-based filtering. We extensively analyzed the proposed framework from various aspects to show how each component, e.g., threshold optimization and profile updating, can effect the performance. The multi-publisher experiments also investigated the performance of the proposed framework in multi-publisher environments. We showed that multi-publisher recommender systems can outperform individual single-publisher recommender systems, significantly.

This research opens up a number of new research directions towards the study of recommender systems for multi-publisher environments. We intend to develop multi-publisher frameworks for other recommendation approaches, such as collaborative, demographic, and hybrid filtering, as well. For example, a multi-publisher collaborative recommender system should fundamentally differ from the one presented in this paper. A possible solution could be training a collaborative filtering model on the publisher's data in the server side, and training a collaborative filtering model on the user's data came from multiple publishers in the client side. Complementary information can be captured from client- and server-side models. A privacy-preserving mechanism to transfer such complementary information from clients to the servers can lead to a multi-publisher collaborative recommender system.

Studying multi-publisher recommender systems for mobile devices is also an interesting future direction. In addition, the proposed content-based filtering approach can be further improved. For instance, the proposed threshold optimization algorithm only uses the pattern of retrieving relevant documents in order to adjust the threshold value. To improve the filtering performance in future, applying the score of retrieved documents for adjusting the dissemination threshold value can be taken into account. In addition, in the proposed sketching technique, the *error* value is set to a constant number. Estimating the *error* value according to the received documents, the users' history, and the users' feedback, can be focused in the future. Moreover, proposing a more accurate estimation of term probabilities in the private profiles can improve the recommendation performance, significantly. Moreover, the proposed language modeling method can be further modified for recommending documents in different languages (i.e., multi-lingual recommender systems). Studying the topic drift problem in dynamic domains would be also an interesting future work for content-based recommender systems.

Acknowledgements We would like to gratefully thank the anonymous reviewers for their invaluable suggestions. We would also like to thank Milad Nasr, Shamim Taheri, and Mostafa Dehghani for their constructive comments towards improving the paper, and Avi Arampatzis for the helpful discussions on the KUN

method. This research was supported in part by a grant from the Institute for Research in Fundamental Sciences (No. CS1396-4-51). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transaction on Knowledge and Data Engineering*, *17*(6), 734–749.

Allan, J. (1996). Incremental relevance feedback for information filtering. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '96* (pp. 270–278). New York, NY.

Antiqueira, L., Oliveira, O. N., Costa, L. D. F., & Nunes, M. D. G. V. (2009). A complex network approach to text summarization. *Information Sciences*, *179*(5), 584–599.

Arampatzis, A. (2001). Unbiased S-D threshold optimization, initial query degradation, decay, and incrementality, for adaptive document filtering. In *Proceedings of tenth text REtrieval conference, TREC-10*. National Institute of Standards and Technology, Special Publication.

Arampatzis, A., Beney, J., Koster, C. H. A., & van der Weide, T. (2000). KUN on the TREC-9 filtering track: Incrementality, decay, and threshold optimization for adaptive filtering systems. In *Proceedings of ninth text REtrieval conference, TREC-9*. National Institute of Standards and Technology, Special Publication.

Arampatzis, A., & van Hameran, A. (2001). The score-distributional threshold optimization for adaptive binary classification tasks. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '01* (pp. 285–293). New York, NY: ACM.

Arampatzis, A., Kamps, J., & Robertson, S. (2009). Where to stop reading a ranked list?: Threshold optimization using truncated score distributions. In *Proceedings of the 32Nd international ACM SIGIR conference on research and development in information retrieval, SIGIR '09* (pp. 524–531). New York, NY: ACM.

Barragáns-Martínez, A. B., Costa-Montenegro, E., Burguillo, J. C., Rey-López, M., Mikic-Fonte, F. A., & Peleteiro, A. (2010). A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences*, *180*(22), 4290–4311.

Beel, J., Langer, S., Nrnberger, A., & Genzmehr, M. (2013). The impact of demographics (age and gender) and other user-characteristics on evaluating recommender systems. In *Proceedings of the 2013 international conference on theory and practice of digital libraries, TPDL '13* (pp. 396–400). Springer: Berlin.

Belkin, N. J., & Croft, W. B. (1992). Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, *35*(12), 29–38.

Besançon, R., Chaudiron, S., Mostefa, D., Timimi, I., & Choukri, K. (2008). The INFILE project: A crosslingual filtering systems evaluation campaign. In *Proceedings of the sixth international conference on language resources and evaluation, LREC '08*. ELRA: Marrakech.

Besançon, R., Chaudiron, S., Mostefa, D., Timimi, I., Choukri, K., & Laïb, M. (2009). Information filtering evaluation: Overview of CLEF 2009 INFILE track. In *Proceedings of the 10th cross-language evaluation forum conference on multilingual information access evaluation: Text retrieval experiments, CLEF '09* (pp. 342–353). Springer: Berlin.

Bobadilla, J., Ortega, F., Hernando, A., & GutiéRrez, A. (2013). Recommender systems survey. *Knowledge-Based System*, *46*, 109–132.

Bogers, T., & van den Bosch, A. (2007). Comparing and evaluating information retrieval algorithms for news recommendation. In *Proceedings of the 2007 ACM conference on recommender systems, RecSys '07* (pp. 141–144). New York, NY: ACM.

Bogers, T., & van den Bosch, A. (2009). Collaborative and content-based filtering for item recommendation on social bookmarking websites. In *Proceedings of the 2009 workshop on recommender systems and the social web, RSWEB '09*.

Bonchi, F., & Ferrari, E. (2010). *Privacy-aware knowledge discovery: Novel applications and new techniques* (1st ed., pp. 369–391). Boca Raton, FL: CRC Press, Inc.

Brandow, R., Mitze, K., & Rau, L. F. (1995). Automatic condensation of electronic publications by sentence selection. *Information on Processing and Management*, *31*(5), 675–685.

Braunhofer, M., Elahi, M., & Ricci, F. (2014). Usability assessment of a context-aware and personality-based mobile recommender system. In *International conference on electronic commerce and web technologies, EC-Web '14* (pp. 77–88). Springer: Berlin.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction.*, *12*(4), 331–370.

Cantador, I., Fernández-Tobías, I., Berkovsky, S., & Cremonesi, P. (2015). Cross-domain recommender systems. In R. Francesco, L. Rokach & B. Shapira (Eds.), *Recommender system handbook* (pp. 919–959). Boston, MA: Springer. https://doi.org/10.1007/978-1-4899-7637-6_27

Castro, J., Rodriguez, R. M., & Barranco, M. J. (2014). Weighting of features in content-based filtering with entropy and dependence measures. *International Journal of Computational Intelligence Systems*, *7*(1), 80–89.

Chen, W., Niu, Z., Zhao, X., & Li, Y. (2014). A hybrid recommendation algorithm adapted in E-learning environments. *World Wide Web*, *17*(2), 271–284.

Choi, K., Yoo, D., Kim, G., & Suh, Y. (2012). A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. *Electronic Commerce Research and Applications*, *11*(4), 309–317.

Cissée, R., & Albayrak, S. (2007). An agent-based approach for privacy-preserving recommender systems. In *Proceedings of the 6th ACM international joint conference on autonomous agents and multiagent systems, AAMAS '07* (pp. 182:1–182:8). New York, NY: ACM.

Cormen, T. H., Stein, C., Rivest, R. L., & Leiserson, C. E. (2001). *Introduction to algorithms* (2nd ed.). New York: McGraw-Hill Higher Education.

Cremonesi, P., Tripodi, A., & Turrin, R. (2011). Cross-domain recommender systems. In *Proceedings of the 2011 IEEE 11th international conference on data mining workshops, ICDM '11* (pp. 496–503). IEEE

Crestani, F., Mizzaro, S., & Scagnetto, I. (2017). *Mobile information retrieval. Springer briefs in computer science*. Berlin: Springer.

Dean-Hall, A., Clarke, C. L. A., Kamps, J., Thomas, P., & Voorhees, E. M. (2012). Overview of the TREC 2012 contextual suggestion track. In *Proceedings of the twenty-first text REtrieval conference, TREC '12*. National Institute of Standards and Technology, Special Publication.

Dean-Hall, A., Clarke, C. L. A., Kamps, J., Thomas, P., & Voorhees, E. M. (2014). Overview of the TREC 2014 contextual suggestion track. In *Proceedings of the twenty-third text REtrieval conference, TREC '14*. National Institute of Standards and Technology, Special Publication.

Dean-Hall, A., Clarke, C. L. A., Kiseleva, J., Thomas, P., & Voorhees, E. M. (2015). Overview of the TREC 2015 contextual suggestion track. In *Proceedings of the twenty-fourth text REtrieval conference, TREC '15*. National Institute of Standards and Technology, Special Publication.

Dean-Hall, A., Clarke, C. L. A., Simone, N., Kamps, J., Thomas, P., & Voorhees, E. M. (2013). Overview of the TREC 2013 contextual suggestion track. In *Proceedings of the twenty-second text REtrieval conference, TREC '13*. National Institute of Standards and Technology, Special Publication.

Elahi, M., Ge, M., Ricci, F., Fernández-Tobías, I., Berkovsky, S., & Massimo, D. (2015). Interaction design in a mobile food recommender system. In *Proceedings of the joint workshop on interfaces and human decision making for recommender systems, IntRS@RecSys '15* (pp. 49–52).

Erkin, Z., Beye, M., Veugen, T., & Lagendijk, R. L. Privacy-preserving content-based recommender system. In *Proceedings of the 14th ACM workshop on multimedia and security, MM&Sec '12* (pp. 77–84). New York, NY: ACM.

Gavalas, D., Konstantopoulos, C., Mastakas, K., & Pantziou, G. (2014). Review: Mobile recommender systems in tourism. *Journal of Network Computer Applications*, *39*, 319–333.

Hanani, U., Shapira, B., & Shoval, P. (2001). Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction*, *11*(3), 203–259.

Hashemi, H. B., Shakery, A., & Faili, H. (2010). Creating a persian-english comparable corpus. In *Proceedings of the 2010 international conference on multilingual and multimodal information access evaluation: Cross-language evaluation forum, CLEF '10* (pp. 27–39). Berlin: Springer.

Hashemi, S. H., Clarke, C. L. A., Kamps, J., Kiseleva, J., & Voorhees, E. M. (2016). Overview of the TREC 2016 contextual suggestion track. In *Proceedings of the twenty-fifth text REtrieval conference, TREC '16*. National Institute of Standards and Technology, Special Publication.

Hersh, W., Buckley, C., Leone, T. J., & Hickam, D. (1994). OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '94* (pp. 192–201). New York, NY: Springer.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transaction on Information Systems*, *22*(1), 89–115.

Hu, R., Dou, W., & Liu, J. (2014). ClubCF: A clustering-based collaborative filtering approach for big data application. *IEEE Transactions on Emerging Topics in Computing*, *2*(3), 302–313.

Hull, D. A. (1997). The TREC-6 filtering track: Description and analysis. In *Proceedings of eighth text REtrieval conference, TREC-6*. National Institute of Standards and Technology, Special Publication.

Hull, D. A. (1998). The TREC-7 filtering track: Description and analysis. In *Proceedings of eighth text REtrieval conference, TREC-7*. National Institute of Standards and Technology, Special Publication.

Hull, D. A., & Robertson, S. E. (1999). The TREC-8 filtering track final report. In *Proceedings of eighth text REtrieval conference, TREC-8*. National Institute of Standards and Technology, Special Publication.

Jurafsky, D., & Martin, J. H. (2009). Speech and language processing: An introduction to natural language processing, computational linguistics and speech recognition. (2nd ed.). [pearson international edition] edn. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, Englewood Cliffs, NJ.

Koren, Y., & Bell, R. (2011). Recommender systems handbook. In R. Francesco, L. Rokach, B. Shapira & P. B. Kantor (Eds.), *Advances in collaborative filtering* (pp. 145–186). Boston, MA: Springer.

Krulwich, B. (1997). LIFESTYLE FINDER: Intelligent user profiling using large-scale demographic data. *AI Magazine*, *18*(2), 37–45.

Kurose, J. F., & Ross, K. W. (2009). *Computer networking: A top-down approach* (5th ed.). Boston: Addison-Wesley Publishing Company.

Lafferty, J., & Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '01* (pp. 111–119). New York, NY: ACM.

Lavrenko, V., & Croft, W. B. (2001). Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '01* (pp. 120–127). New York, NY: ACM.

Lavrenko, V., Schmill, M., Lawrie, D., Ogilvie, P., Jensen, D., & Allan, J. (2000). Language models for financial news recommendation. In *Proceedings of the ninth international conference on information and knowledge management, CIKM '00* (pp. 389–396). New York, NY: ACM.

Li, P., Jiang, J., & Wang, Y. (2010). Generating templates of entity summaries with an entity-aspect model and pattern mining. In *Proceedings of the 48th annual meeting of the association for computational linguistics, ACL '10* (pp. 640–649). Stroudsburg, PA.

Linqi, G., & Li, C. (2008). Hybrid personalized recommended model based on genetic algorithm. In *Proceedings of the 4th IEEE international conference on wireless communications, networking and mobile computing, 2008, WiCOM '08* (pp. 1–4). IEEE.

Liu, X., & Croft, W. B. (2004). Cluster-based retrieval using language models. In: *Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '04* (pp. 186–193). New York, NY: ACM.

Lommatzsch, A., Kille, B., Hopfgartner, F., Larson, M., Brodt, T., Seiler, J., & Özgöbek, Ö. (2017). CLEF 2017 NewsREEL Overview: A stream-based recommender task for evaluation and education. In *Proceedings of the 8th international conference of the CLEF association: Experimental IR meets multilinguality, multimodality, and interaction, CLEF '17* (pp. 239–254). Berlin: Springer.

Lops, P., de Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In R. Francesco, L. Rokach, B. Shapira & P. B. Kantor (Eds.), *Recommender Systems Handbook* (pp. 73–105). Berlin: Springer.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Developments*, *2*(2), 159–165.

Lv, Y., & Zhai, C. (2009). A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM conference on information and knowledge management, CIKM '09* (pp. 1895–1898). New York, NY: ACM.

Maidel, V., Shoval, P., Shapira, B., Taieb-Maimon, M. (2008). Evaluation of an ontology-content based filtering method for a personalized newspaper. In *Proceedings of the 2008 ACM conference on recommender systems, RecSys '08* (pp. 91–98). New York, NY: ACM.

Mani, I. (1999). *Advances in automatic text summarization*. Cambridge, MA: MIT Press.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York, NY: Cambridge University Press.

Montazeralghaem, A., Zamani, H., & Shakery, A. (2016). Axiomatic analysis for improving the log-logistic feedback model. In *Proceedings of the 39th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '16* (pp. 765–768). New York, NY: ACM.

Montazeralghaem, A., Zamani, H., & Shakery, A. (2016). Cross domain user engagement evaluation. In *Proceedings of the 38th European conference on information retrieval, ECIR '16* (pp. 754–760). Berlin: Springer.

Mooney, R. J., & Roy, L. (2000). Content-based Book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on digital libraries, DL '00* (pp. 195–204). New York, NY: ACM.

Müller, W., Eisenhardt, M., & Henrich, A. (2005). Scalable summary based retrieval in P2P networks. In *Proceedings of the 14th ACM international conference on information and knowledge management, CIKM '05* (pp. 586–593). New York, NY: ACM.

Neto, J. L., Santos, A. D., Kaestner, C. A., Alexandre, N., Santos, D., A, C. A., Alex, K., Freitas, A. A., & Parana, C. (2000). Document clustering and text summarization. In *Proceedings of the 4th international conference practical applications of knowledge discovery and data mining* (pp. 41–55).

Parameswaran, R., & Blough, D. M. (2007). Privacy preserving collaborative filtering using data obfuscation. In *IEEE international conference on granular computing, GRC '07* (pp. 380–380). IEEE.

Parapar, J., Bellogín, A., Castells, P., & Barreiro, A. (2013). Relevance-based language modelling for recommender systems. *Information Processing and Management*, *49*(4), 966–980.

Pazzani, M., & Billsus, D. (2007). Content-based recommendation systems. In *Lecture notes in computer science the adaptive web* (Vol. 4321, pp. 325–341). Berlin: Springer.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, *13*(5–6), 393–408.

Polat, H., & Du, W. (2003). Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proceedings of the third IEEE international conference on data mining, ICDM '03* (pp. 625–639). IEEE.

Polatidis, N., Georgiadis, C. K. (2014). Mobile recommender systems: An overview of technologies and challenges. *CoRR*. http://arxiv.org/abs/1408.6930

Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '98* (pp. 275–281). New York, NY: ACM.

Rahmatizadeh Zagheli, H., Zamani, H., & Shakery, A. (2017). A semantic-aware profile updating model for text recommendation. In *Proceedings of the eleventh ACM conference on recommender systems, RecSys '17* (pp. 316–320). New York, NY: ACM.

Ricci, F. (2010). Mobile Recommender Systems. *Journal of IT and Tourism*, *12*(3), 205–231.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). *Recommender systems handbook* (1st ed.). New York, NY: Springer.

Robertson, S., & Hull, D. A.(2000). The TREC-9 filtering track final report. In *Proceedings of ninth text REtrieval conference, TREC-9* (pp. 25–40). National Institute of Standards and Technology, Special Publication.

Robertson, S., & Soboro, I. (2001). The TREC 2001 filtering track report. In *Proceedings of tenth text REtrieval conference, TREC '01* (pp. 26–37). National Institute of Standards and Technology, Special Publication.

Robertson, S., & Soboroff, I. (2002). The TREC 2002 filtering track report. In *Proceedings of eleventh text REtrieval conference, TREC '02*. National Institute of Standards and Technology, Special Publication.

Rocchio, J. J. (1971). *Relevance feedback in information retrieval*. New Jersey: Prentice Hall.

Saggion, H., & Lapalme, G. (2002). Generating indicative-informative summaries with sumUM. *Computational Linguistics*, *28*(4), 497–526.

Saggion, H., & Poibeau, T. (2012). Automatic text summarization: Past, present and future. In *Multisource, multilingual information extraction and summarization, theory and applications of natural language processing* (pp. 3–13). Berlin: Springer.

Sakai, T., & Spärck-Jones, K. (2001). Generic summaries for indexing in information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '01* (pp. 190–198). New York, NY: ACM.

Shinde, S. K., & Kulkarni, U. (2012). Hybrid personalized recommender system using centering-bunching based clustering algorithm. *Expert System Applications*, *39*(1), 1381–1387.

Soboroff, & I., Nicholas, C. (2000). Collaborative filtering and the generalized vector space model. In *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '00* (pp. 351–353). New York, NY: ACM.

Song, F., & Croft, W. B. (1999). A general language model for information retrieval. In *Proceedings of the eighth international conference on information and knowledge management, CIKM '99* (pp. 316–321). New York, NY: ACM.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, *4*, 1–19.

Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2008). Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the 2008 ACM conference on recommender systems, RecSys '08* (pp. 267–274). New York, NY: ACM.

Tang, J., Wu, S., Sun, J., & Su, H. (2012). Cross-domain collaboration recommendation. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '12* (pp. 1285–1293). New York, NY: ACM.

Tryfonopoulos, C., & Andreescu, L. (2011). Pricing information goods in distributed agent-based information filtering. *Lecture notes in computer science on the move to meaningful internet systems: OTM 2011* (Vol. 7044, pp. 163–181). Berlin: Springer.

Tryfonopoulos, C., Idreos, S., Koubarakis, M., & Raftopoulou, P. (2014). Distributed large-scale information filtering. *Transactions on Large-Scale Data- and Knowledge Centered Systems*, *13*, 91–122.

Tryfonopoulos, C., Koubarakis, M., & Drougas, Y. (2009). Information filtering and query indexing for an information retrieval model. *ACM Transactions on Information Systems*, *27*(2), 10:1–10:47.

Turney, P. D., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, *37*(1), 141–188.

van Metern, R., & van Someren, M. (2002). *Using content-based filtering for recommendation*. Foundation for Research and Technology - Hellas. Tech. rep.

Wan, X., & Xiao, J. (2010). Exploiting neighborhood knowledge for single document summarization and keyphrase extraction. *ACM Transactions on Information Systems*, *28*(2), 8:1–8:34.

Wang, J., Li, Q., Chen, Y. P., Liu, J., Zhang, C., & Lin, Z. (2010). News recommendation in forum-based social media. In *Proceedings of AAAI conference on artificial intelligence*. AAAI Press.

Wang, Q., Cao, W., & Liu, Y. (2014). A novel clustering based collaborative filtering recommendation system algorithm. In *Embedded and multimedia for human-centric computing, lecture notes in electrical engineering advanced technologies* (Vol. 260, pp. 673–680). Netherlands: Springer.

Wang, S., Sun, J., Gao, B. J., & Ma, J. (2012). Adapting vector space model to ranking-based collaborative filtering. In *Proceedings of the 21st ACM international conference on information and knowledge management, CIKM '12* (pp. 1487–1491). New York, NY: ACM.

Yang, B., & Jeh, G. (2006). Retroactive answering of search queries. In *Proceedings of the 15th International Conference on World Wide Web, WWW '06* (pp. 457–466). New York, NY: ACM.

Yang, X., Guo, Y., Liu, Y., & Steck, H. (2014). A survey of collaborative filtering based social recommender systems. *Computer Communications*, *41*, 1–10.

Zagheli, H. R., Ariannezhad, M., & Shakery, A. (2017). Negative feedback in the language modeling framework for text recommendation. In *Proceedings of the 39th European conference on information retrieval, ECIR* (pp. 662–668).

Zamani, H., & Croft, W. B. (2016). Embedding-based query language models. In *Proceedings of the 2016 ACM international conference on the theory of information retrieval, ICTIR '16* (pp. 147–156). New York, NY: ACM. https://doi.org/10.1145/2970398.2970405.

Zamani, H., & Croft, W. B. (2017). Relevance-based word embedding. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, SIGIR '17* (pp. 505–514). New York, NY: ACM.

Zamani, H., Dadashkarimi, J., Shakery, A., & Croft, W. B. (2016). Pseudo-relevance feedback based on matrix factorization. In *Proceedings of the 25th ACM international on conference on information and knowledge management, CIKM '16* (pp. 1483–1492). New York, NY: ACM.

Zamani, H., Moradi, P., & Shakery, A. (2015). Adaptive user engagement evaluation via multi-task learning. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, SIGIR '15* (pp. 1011–1014). New York, NY: ACM.

Zhai, C. (2008). Statistical language models for information retrieval a critical review. *Foundation on Trends in Information Retrievel*, *2*(3), 137–213.

Zhai, C., Jansen, P., & Evans, D. A. (2000). Exploration of a heuristic approach to threshold learning in adaptive filtering. In *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '00* (pp. 360–362). New York, NY: ACM.

Zhai, C., & Lafferty, J. (2001). Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on information and knowledge management, CIKM '01* (pp. 403–410). New York, NY: ACM.

Zhai, C., & Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, *22*(2), 179–214.

Zhan, J., Hsieh, C. L., Wang, I. C., Hsu, T. S., Liau, C. J., & Wang, D. W. (2010). Privacy-preserving collaborative recommender systems. *Transactions on Systems, Man, and Cybernetics, Part C*, *40*(4), 472–476.

Zhang, Y., & Callan, J. (2001). The bias problem and language models in adaptive filtering. In *Proceedings of tenth text REtrieval conference, TREC-10*. National Institute of Standards and Technology, Special Publication.

Zhang, Y., & Callan, J. (2001). Maximum likelihood estimation for filtering thresholds. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '01* (pp. 294–302). New York, NY: ACM.

Zhang, Y., & Callan, J. (2003). An unbiased generative model for setting dissemination thresholds. In W. B. Croft & J. Lafferty (Eds.), *Language modeling for information retrieval* (pp. 189–217). Netherlands: Springer.

Zhang, Y., Cao, B., & Yeung, D. Y. (2010). Multi-domain collaborative filtering. In *Proceedings of the twenty-sixth conference on uncertainty in artificial intelligence, UAI'10* (pp. 725–732). Arlington: AUAI Press.

Zimmer, C., Tryfonopoulos, C., Berberich, K., Koubarakis, M., Weikum, G. (2008). Approximate information filtering in peer-to-peer networks. In *Proceedings of the 9th international conference on web information systems engineering, WISE '08* (pp. 6–19). Berlin: Springer.