A personal data store approach for recommender systems: enhancing privacy without sacrificing accuracy

Itzik Mazeh, Erez Shmueli

Department of Industrial Engineering, Tel-Aviv University, Tel-Aviv, Israel

ARTICLE INFO

Article history:
Received 24 February 2019
Revised 24 June 2019
Accepted 1 August 2019
Available online 8 August 2019

Keywords:
Recommender systems
Privacy
Personal data stores

ABSTRACT

Recommender systems have become extremely common in recent years, and are applied in a variety of domains. Existing recommender systems exhibit two major limitations: (1) Privacy - each service provider holds a database that contains information about all of its users; and (2) Partial view - when recommending to users, each such service can rely only on data that were collected by the service itself.

The Open Personal Data Store (openPDS) architecture was recently suggested for storing personal data in a privacy preserving way. Inspired by openPDS, we suggest a novel architecture for recommender systems that overcomes the two limitations mentioned above. The suggested architecture allows the recommender system to utilize rich data collected about the user (possibly through other services) to produce more accurate recommendations, while allowing its users to manage and gain control over their own data.

We evaluate the suggested architecture on two different use cases: movies and web browsing, and compare its performance with that of a popular non-privacy-aware collaborative-filtering algorithm. We find that in comparison to the alternative approach, our approach is able to enhance privacy significantly without sacrificing the accuracy level of the recommendations (and in some cases providing even higher level of accuracy).

© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

Recommender systems are a class of information filter systems, whose main goal is to provide personalized recommendations, content, and services to users. Recommender systems typically help users find products, such as movies, books, articles, news items and others, that fit their personal preferences and needs. Millions of people around the world interact with a wide variety of recommender systems on a daily basis. Such systems are responsible for offering users content, which might be interesting to them, relying on some inherent recommendation mechanism.

Recommender systems have long been an area of interest and research, dating back to the first published papers on collaborative filtering in the 1990s (Hill, Stead, Rosenstein, & Furnas, 1995; Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994; Shardanand & Maes, 1995). The importance of recommender systems is rising constantly, due to their crucial contribution to personalized services, which is almost a mandatory attribute of information systems nowadays. For this reason, this field is in persistent quest for improvements and extensions (Adomavicius & Tuzhilin, 2005).

In order to generate a recommendation for a given user, the system must have some gathered data about the user, ranging from past behavior within the system to personal data about the user. On one hand, in the vast majority of cases, the data about the user which are used by the service provider, are stored and kept by the service provider. This implies the inherent privacy challenge which arises when dealing with recommender systems. Moreover, these privacy concerns are multiplied nowadays, given that an average user interacts with dozens of service providers. In such a setting, the user effectively loses control over her data. On the other hand, from the service provider’s perspective, the service provided to the user can potentially be improved, if the recommendations for the user were based on a more rich, diverse data about the user, exceeding the data possessed by the given service provider.

The Open Personal Data Store (openPDS) architecture was recently suggested for storing personal data in a privacy preserving way (de Montjoye, Shmueli, Wang, & Pentland, 2014). In openPDS, the user’s data is stored in a central location, managed and controlled by the user. This architecture offers a significant improvement in the privacy provided to the user. Additionally, the user is able to grant - or not, according to her will - each ser-
vice provider with access to different data pieces within the data store.

In the scope of this work, we suggest an architecture for a recommender system which is compatible with the underlying principles of openPDS. Since in the openPDS setting the user’s data are stored and managed at a single location, without involving other users’ data, we propose in this work a content/context-based recommender system which exploits data about the given user only (as opposed to collaborative-filtering systems that exploit data about a community of users).

We evaluate the suggested model using two different datasets (web browsing data and the MovieLens dataset), which originate from two separate realms and consist of different properties. For the evaluation purposes, the model ranks a hold-out set of items and we evaluate the model’s output using the users’ actual preferences. The proposed model results are compared to a popular collaborative filtering algorithm that is not privacy-aware.

We find that the suggested approach is able to enhance privacy without sacrificing the accuracy of the recommendations (and in some cases even to provide higher accuracy), in comparison to the alternative approach. Moreover, we find that exploiting data from multiple service providers, which is an inherent property of the suggested architecture, plays a significant role in its ability to provide highly accurate recommendations.

This work continues from hereon as follows. In Section 2 relevant background is provided, including a basic introduction to recommender systems, user modeling and privacy. In Section 3 we describe the proposed method, including a thorough description of the openPDS architecture and a formulation of the suggested model. Section 4 describes in detail the datasets which were used. Section 5 discusses the evaluation of the model, and Section 6 concludes the findings of this work and discusses some future research directions.

2. Related work

2.1. Recommender systems

Recommender systems are meant to aid users to obtain personalized solutions to the specific problems where they are applied (Resnick & Varian, 1997). On one hand, they are creating an easier and better user experience where they are deployed, especially due to the information overload problem (Epller & Mengis, 2004). On the other hand they are helping the recommendations provider to achieve various goals such as increased conversion rates and sales, increased user engagement and satisfaction, etc. (Symeonidis, Nanopoulos, Papadopoulos, & Manolopoulos, 2008). Indeed, recommender systems were shown to be imperative for business success in various settings (Al-Shamri, 2016).

The basic challenge of every recommender system is to predict the level of interest a certain user will have in a specific item she has not rated yet. More specifically, this challenge might take the form of either predicting the rating the user will give the unrated item, or by assigning a score to each unrated item and then ranking these items according to the assigned scores, from the most relevant to the least relevant. In the scope of this work, we focus on the task of ranking a set of unrated items. The already known ratings, which are the basis to any prediction procedure, are obtained through observations of the interactions between users and items. These interactions can be either positive or negative, and can be transmitted both explicitly and implicitly.

The relevance estimation of not-yet-rated items can be made in a variety of methods, based on different approaches, models and points of view. Recommender systems are usually divided into sub-classes, according to the recommendation generation process that is performed behind the scenes. We can generalize the different recommendation algorithms by stating that the relevance for a not-yet-rated item is estimated by using data about the users, about the items, or both.

At a high level, recommender systems are divided into two main sub-classes: content-based recommender systems and collaborative filtering recommender systems (Balabanović & Shoham, 1997). In content-based recommender systems, the relevance of a not-yet-rated item i for a certain user u is estimated based on the ratings given solely by u to items which are similar to i. In contrast, in collaborative filtering (CF) recommender systems, u’s estimated relevance for i is based on ratings given by other users as well (Lang, 1995; Mooney & Roy, 2000; Pazzani & Billsus, 1997).

2.2. Privacy

While personal data about individuals (e.g., location data, browsing behavior, personal preferences, etc.) could be very useful, it also poses considerable threats to their privacy (Wang, Zheng, Jiang, & Ren, 2018). Leakage of personal data, misuse of it, or attacks by malicious users are commonplace (Tsay-Vogel, Shanahan, & Signorielli, 2018).

Data privacy legislation has been evolving along with personal data collection and processing technologies over the years (Raab & Szekely, 2017). Older examples in Europe include Article 8 of the Charter of Fundamental Rights of the European Union (EU) which guarantees a citizen the right to the protection of personal data (van Loenen, Kulk, & Ploeger, 2016), and the Data Protection Directive introduced in 1995 by the EU (DIR95), which was meant to safeguard citizens’ privacy for a misuse or unnecessary collection (Ryz & Grest, 2016). On April 2016, the EU Parliament approved the EU General Data Protection Regulation (GDPR). GDPR has introduced several changes compared to DIR95, presenting new definitions and principles and clarifying previous ones. Computers becoming commonplace, the rise of smartphones and social media and organizational dependence on digital mediums have all been disregarded by DIR95 – largely because they did not exist in the same way when the act was formulated. The new regulations introduced tougher fines for non-compliance and breaches and gave people more say over what companies can do with their data. As a concrete example, every data subject now has the right to be forgotten, essentially meaning that the service provider has to delete the personal data of the data subject upon request. As another example, every data subject now has the right to data portability, meaning that the service provider has to provide a copy of the personal data of the data subject upon request.

Since their introduction, numerous papers were written aiming at explaining what the new regulations mean (e.g., Beckett, 2017; Tikkinen-Piri, Rohunen, & Markkula, 2018) and providing guidelines on how to adapt information systems in general and recommender systems in particular (e.g., Tejeda-Lorente, Bernabé-Moreno, Herce-Zelaya, Porcel, & Herrera-Viedma, 2018) to the new regulations. In this paper, we focus on suggesting a technological mechanism for privacy that can aid in supporting some of the requirements mentioned in the new regulations (such as the right to be forgotten and the right to data portability) in the context of recommender systems.

2.3. Privacy preserving recommender systems

Since the collection of users’ personal data for generating recommendations is an essential part of every recommender systems, various privacy concerns arise. In fact, recommender systems not only gain access to personal data of users, but may also share this data with third parties (Enck et al., 2014), expose this data
to the public via open APIs of the service providers (Wondracek, Holz, Kirda, & Kruegel, 2010), or be vulnerable to attacks by malicious users (Li, Lv, Shang, & Gu, 2014; Yuan, Chen, & Yu, 2010).

Privacy preserving collaborative filtering (PPCF) (Casino, Pat-sakis, Puig, & Solanas, 2013; Friedman, Knijnenburg, Vanhecke, Martens, & Berkovsky, 2015), enables the practice of collaborative filtering without leaking private information to the recommendation server - in the case of a centralized setting, or to other collaborating parties - in the case of a distributed setting. Existing works on PPCF can be broadly categorized into two ar- eas: (1) Perturbation-based solutions and (2) Cryptography-based solutions.

Perturbation-based methods rely on adding noise to the pri- vate data of users before sending them to the recommendation server, with the goal of enhancing privacy. Preliminary studies in this category were based on adding random noise and include for example, Polat and Du (2003), Polat and Du (2005b), Zhang, Ford, and Makedon (2006b) and Berkovsky, Eytami, Kuflik, and Ricci (2007). Besides reducing the level of recommendation accuracy due to the addition of noise, such methods were shown to be inse- cure, since the server has a very high chance to reconstruct the true ratings of the users (Aggarwal, 2007; Huang, Du, & Chen, 2005; Zhang, Ford, & Makedon, 2006a). Shokri, Pedarsani, Theodor- akopoulos, and Hubaux (2009) proposed to preserve users' privacy by obfuscating the user-item connections among similar users be- fore sending the users' data to the recommendation server. While this type of obfuscation may hide users' interests in particular items, it may still reveal users' interests at a higher level, as ex-plained by Li, Lv, Shang, and Gu (2017). McSherry and Mironov (2009) and Machanavojhala, Korolova, and Sarma (2011) suggested the use of differential privacy for PPCF. However, besides the afore-mentioned effect of adding noise on accuracy, their methods were also found to be vulnerable to KNN attacks (Zhu, Ren, Zhou, Rong, & Xiong, 2014). Zhu, Li, Ren, Zhou, and Xiong (2013) proposed a differential privacy scheme which is not vulnerable to that at- tack. However, the proposed scheme was unable to maintain a good trade-off between privacy and accuracy. Chow, Pathak, and Wang (2012) proposed a PPCF method in which users are clus- tered and recommendations are generated based on the ratings of similar users in the same cluster. Casino, Domingo-Ferrer, Pat- sakis, Puig, and Solanas (2015) proposed a PPCF method based on micro-aggregation, in which k-anonymity can be guaranteed. Wei, Tian, and Shen (2018) proposed an improvement to the k-anonymity based PPCF method, namely (p, l, α)-diversity, which takes into account the attacker's prior knowledge about users' rat- ings (p) and enforces a certain diversity level (l, α) among users in each group. While in the three studies above, better privacy pro- tection is achieved in comparison to adding random noise, tradeoff still exists between accuracy and privacy.

Cryptography-based solutions apply secure cryptographic prim- itives on the private data, aiming to avoid the disclosure of sen- sitive information to the recommendation server. Canny (2002), Miller, Konstan, and Riedl (2004) and Erkin, Beye, Veugen, and La- gendijk (2010) proposed PPCF methods which use secure multi- party computation (SMC) and homomorphic encryption protocols. The main drawback of these studies is the increased communica- tion and computational costs induced by the cryptographic mecha- nisms. In addition, if two or more parties decide to collude, pri- vacy may be compromised severely. Badsha, Yi, Khalil, and Bertino (2017) suggested a user-based PPCF method which uses the Boneh, Goh and Nissim (BGN) cryptosystem. The unique properties of the BGN cryptosystem allow their method to compute a multiplication operation over encrypted data using a single server, thereby improving the level of privacy. However, the computational time of their method is relatively high (i.e., the time complexity of a sin- gle recommendation is O(#users · #items)), and it shares the same privacy concern due to potential collusion. Erkin, Veugen, Toft, and Lagendijk (2012) suggested the use of data packing in PPCF to min- imize computation and communication overhead. However, this method still shares the same privacy concern if the recommenda- tion server colludes with the decryption server. Li et al. (2017) pro- posed a ppf method for users in online social communities, in which users are organized into groups with diverse interests and interact with the recommender server via interest-specific pseudo users, so that individual user's personal interest information re- mains hidden from the server. However, the grouping of users may hinder the full potential of a fully personalized recommender sys- tem. Several other studies, such as Polat and Du (2005a), Polat and Du (2005a), Armknecht and Struve (2011), Basu, Vaidya, and Kikuchi (2011), Jeckmans, Tang, and Hartel (2012), Basu, Vaidya, Kikuchi, and Dimitrakos (2013), Boutet, Frey, Guerranou, Jegou, and Kermarrec (2016), and Shmueli and Tassa (2017), have investigated the use of cryptography for PPCF in distributed settings, which are out of scope of this paper.

To summarize this part, the main advantage of perturbation- based techniques is their efficiency since they usually do not in- cur complicated manipulations in order to preserve privacy. How- ever, they generally trade accuracy for privacy, which means users receive lower-quality recommendations in order to protect their privacy. In contrast, cryptography-based solutions rely on differ- ent cryptographic primitives such as homomorphic encryption and secure multiparty computations, and since data is not altered, the same level of accuracy is maintained. However, cryptography- based solutions usually entail high communication and computa- tional costs, which hinder their scalability, especially considering the fact that modern recommender systems may face millions or even billions of users and items. Moreover, despite the fact that they generally provide rigorous privacy guarantees, such guaran- tees typically do not hold in the case of collusion. In this paper, we take an entirely different approach to preserving privacy in recommender systems which relies on the use of personal data stores.

2.4. Personal data stores and openPDS

Several recent studies proposed a novel approach regarding the management, storage and accessibility of personal data (Allard et al., 2010; Bell, 2001; Cassel & Wolz, 2001; Coroama & Langhein- rich, 2006; Gerber et al., 2010; Kirkham, Ravet, Winfield, & Kel- lomäki, 2011; Mulligan & Schwartz, 2000; Want et al., 2002). The new approach suggests storing the user’s data locally at the client side, sometimes referred to as a Personal Data Store, or PDS, rather than on the service provider’s server side. These studies have made a significant step towards a more user focused, privacy preserving architecture, which can bring Westin’s vision of privacy into some- what of fulfillment in the contemporary data-driven era.

In the scope of this work, we are specifically inspired by the architecture of openPDS, proposed by de Montjoye et al. (2014), openPDS suggests a privacy-aware alternative to a reality in which users share their personal data with multiple service providers and, as a result, effectively lose control over their data since they do not know where it is stored, who is accessing it and for what purposes.

The architecture and the data flow of openPDS are illustrated in Fig. 1. The user’s personal data, generated by the different service providers, are stored in her personal data store. When installing each application on her mobile device, the user is asked to grant the application with access to different types of data. The user is of course free to approve or disapprove each such request, and the application will then adapt its operation according to the user-specific granted access.
A key component in the openPDS architecture is SafeAnswers (SA). SafeAnswers is a module inside the openPDS architecture, which enables the service provider to utilize the user's personal data in a privacy-preserving manner. To better explain the way SA works, consider a service provider (e.g., Pandora or Foursquare) who wants to provide the user some personalized content. The service provider will send a request to the user's openPDS. This request will be handled by the service provider's SA module that was pre-installed inside the user's openPDS as follows. The SA module will access the user's personal data it is permitted to. Based on the accessed personal data, the SA module will perform the required computation. Finally, only the result of the computation (not the entire data accessed) will be sent out of openPDS to the service provider, which in turn will generate and return the personalized content to the user.

From the user's privacy perspective, openPDS offers a significant improvement over the alternative common scenario, in which the user's personal data are sent to multiple service providers, and once sent, the user practically loses track and control over these data.

3. The proposed method

As explained above, most recommender systems nowadays rely on a collaborative filtering recommendation mechanism, which inherently exploits data about a community of users in order to generate recommendations. Hence, each service provider (e.g., Amazon, Pandora, eBay, etc.) is required to hold a dataset consisting of data about multiple users. From the user's standpoint, in addition to sending her data to the service provider, and thus effectively losing her ability to track and control it, she also repeats this process with multiple service providers, resulting in a reality in which her data is stored and accessed by multiple parties, even such that she is not aware of. From a privacy perspective, this reality is quite concerning.

Inspired by the openPDS architecture described above, in this work we suggest a recommender system, which remedies the privacy disadvantages mentioned above. As demonstrated in Fig. 1, openPDS enables storing the user's data from various service providers in a central, secure location. This architecture avoids the need of the user to send her data to multiple unsupervised locations. Since the user's data are stored and managed at a single location, without involving other users' data, we propose in this work a content-based recommender system, exploiting solely data of the given user. (As will become clearer later on, such an approach can also be extended to support context-based recommender systems.)

The shift from a collaborative filtering recommender system to a content-based one might result in a downgrade in terms of the quality of the recommendation generated, since the recommendation process is now lacking the option of exploiting other users' data. This downside can be remedied by an additional capability provided to the recommender system by the properties of the openPDS architecture, namely exploiting the user's data from multiple service providers, enabled by openPDS. For example, in order to enrich its understanding of the user and her interests, Amazon will be able, if granted, to access the user's web browsing history as well as her Facebook data, in addition to Amazon's own data about the user, of course.

In the rest of this section, we formulate and describe in detail the proposed model for PDS based recommender systems. In Section 3.1 we describe how things are commonly being done today and in Section 3.2 we describe the suggested alternative.

3.1. Current common setting

As mentioned above, the common approach for implementing recommender systems is based on collaborative filtering. Generally speaking, collaborative filtering recommender systems exploit data from multiple users in order to generate recommendations. This is typically done by exploiting a user-item matrix which contains the users' ratings for the various items in the system.

More formally, we assume a dataset, consisting of the following components and characteristics:

1. A set of $m$ users, denoted by $U = \{u_1, u_2, \ldots, u_m\}$.
2. A set of $p$ service providers, denoted by $S = \{s_1, s_2, \ldots, s_p\}$.
3. Each service provider $s$ offers a set of $q_s$ items, denoted by $I_s = \{i_1, i_2, \ldots, i_{q_s}\}$.
4. Each service provider $s$ stores a user-item rating matrix, $R_s$, which is an $m \times q_s$ matrix where $R_s(x, y)$ is the rating that the user $u_x$ gave to item $i_y \in I_s$, a value which is usually taken from a small range of positive integers, say $\{1, 2, 3, 4, 5\}$, and $R_s(x, y) = 0$ if $u_x$ did not rate $i_y$.

While evaluating our proposed method, we focus on an item-based collaborative filtering (IBCF) algorithm, which is one of the most widely deployed collaborative filtering techniques (Ekstrand, Riedl, Konstan et al., 2011). In comparison to other memory based approaches, it presents significantly better scalability, making it more suitable for deployment in e-commerce sites. While item-based methods have comparable performance to that of model-based methods when predicted ratings are the required output, the former methods tend to show better performance when the prediction task in mind is that of ranking (Ekstrand et al., 2011). Furthermore, item-based recommendations are more easily interpretable to the user.

The formulation of the recommendation mechanism in this case is given in Algorithm 1. We demonstrate the way the algorithm works using a running example based on the user-item rating matrix (of a certain service provider) shown in Table 1. In this example, the recommender system aims at providing Dave a ranked list of the items he has not yet rated (i.e. items 4 and 5), scored from the most relevant to the least relevant.
The IBCF algorithm starts by binarizing the ratings of items that user $u$ has rated. That is, given a rating threshold as input, denoted as $\text{rating\_thresh}$, the algorithm replaces the rating $R(u, i)$ with 1 if $R(u, i) \geq \text{rating\_thresh}$ and $-1$ otherwise. This binarization allows us to interpret $R(u, i) = 1$ as $u$ “likes” $i$, and $R(u, i) = -1$ as $u$ does not like $i$. It is important to note that in some cases, the rating matrix is already given in a binary form (e.g., the user liked or disliked an item), and therefore, it is not required to perform this binarization stage. We then denote the set of items that $u$ liked as $I_u$. In our example, assuming $\text{rating\_thresh} = 3$, we get $\text{I}_u = \{\text{Item}_1, \text{Item}_3\}$.

Next, the IBCF algorithm represents each unrated or liked item $i$ as an $(m-1)$-dimensional vector, denoted as $v(i)$, containing the ratings given to $i$ by all users, excluding $u$. In our example, $v(\text{Item}_4) = < 3, 1, 5 >$.

Then, for each unrated item $i$, the IBCF algorithm calculates the cosine similarity between $v(i)$ and each of the vectors of items that $u$ liked, and computes the sum over these values. In our example, the unrated items are $\text{Item}_4$ and $\text{Item}_5$, and the cosine similarity calculations yield

\[
\begin{align*}
\text{cosine\_similarity}(v(\text{Item}_4), v(\text{Item}_1)) & = 0.932 \\
\text{cosine\_similarity}(v(\text{Item}_4), v(\text{Item}_3)) & = 0.986 \\
\text{cosine\_similarity}(v(\text{Item}_5), v(\text{Item}_1)) & = 0.69 \\
\text{cosine\_similarity}(v(\text{Item}_5), v(\text{Item}_3)) & = 0.711
\end{align*}
\]

and thus
\[
\begin{align*}
\text{score}(\text{Item}_4) & = 0.932 + 0.986 = 1.918 \\
\text{score}(\text{Item}_5) & = 0.69 + 0.711 = 1.401.
\end{align*}
\]

Finally, the IBCF algorithm ranks all unrated items according to their scores, and presents these items to the user according to their ranking. In our example, the algorithm would rank $\text{Item}_4$ higher than $\text{Item}_5$. Intuitively, this implies that among the unrated items, $\text{Item}_4$ had more similar users (which are not Dave) preferences to items in $I_{\text{Dave}}$, than $\text{Item}_5$ did.

It is important to note the following points:

- The cosine similarity measure was shown to perform well in the case of item-to-item collaborative filtering (Ekstrand et al., 2011).
- The summation of similarities was shown to perform well in the case of ranking prediction (Ekstrand et al., 2011).

### 3.2. The proposed alternative

From an architectural perspective, the IBCF recommender system that was presented in the previous subsection, assumes that each service provider $s$, has access to the entire $R_s$ matrix (i.e., the ratings of all users to items that $s$ offers), but it has no access at all to data of other service providers.

As stated above, the proposed architecture which relies on the openPDS architecture, is different in the sense that the user is now the one that stores her rating of all items (even if offered by different service providers) and the service providers do not have access to these data. In that case, the recommendation algorithm is executed in the user’s personal data store. While this difference inherently offers an enhanced level of privacy to the user (for details, the reader is referred to de Montjoye et al., 2014), it also entails that when generating recommendations for a given user, the recommendation algorithm cannot longer access rating data of other users. We therefore suggest a new recommendation algorithm that is more suitable to the PDS case, named PDS-inspired content-based (PDSCB).

In order to generate a recommendation for a given user $u$ to an item offered by a given service provider $s$, the PDSCB proposed model relies solely on $u$’s data. However, rather than using only the data provided by $s$, data from additional service providers is exploited as well. In terms of the dataset available, similarly to the common setting described in the previous subsection, also the PDSCB model assumes $U, S$ and $I_s$, i.e. the sets of users, service providers and items offered by each service provider, respectively. However, it makes use only of $u$’s slices of the rating matrices, i.e. $u$’s row in each rating matrix $R_s$.

The difference between the architectures of the existing IBCF recommender system and the suggested PDSCB recommender system is illustrated in Fig. 2.

As mentioned above, the different items in the IBCF approach are represented as vectors, where each entry represents a specific user’s rating. While the content-based approach also represents items as vectors, the space in which the vectors are built is quite different. To enable the content representation of items, we assume:

1. The content is represented by different terms. Consider an exhaustive set of terms $T = \{\text{term}_1, \text{term}_2, \ldots, \text{term}_n\}$ containing terms from all items in the dataset, where $|T| = n$. $T$ defines an $R^n$ terms-space, where each term $t_d \in T$, is represented in the $n$-dimensional space as the corresponding dimension $d \in \{1, 2, \ldots, n\}$.

2. For each item $i$, some relevance score is assigned for each term $t_d \in T$. Thus, each item $i$ can be represented as: $\text{content}(i) = \langle t_1, t_2, \ldots, t_m \rangle$. Equivalently, each item can be represented as an $n$-dimensional vector $v(i) = \langle v_1, v_2, \ldots, v_n \rangle$, where each entry $v_d$ represents the relevance score of the corresponding term $t_d$ in a given item.

Relevance scores are typically in the range of $(0,1)$. However, in some cases a threshold, denoted as $\text{score\_thresh}$, is used in order to discard terms with scores lower than $\text{score\_thresh}$. The assigned value in $v(i)$ for such terms will then be 0.
Assuming the content-based vector representation of items as described above, the PDSCB algorithm, as the IBCF one, aims at ranking for a user $u$ a set of items, offered by a service provider $s$, she has not rated. The formulation of the PDSCB recommendation mechanism is given in Algorithm 2.

Similarly to the IBCF algorithm, the PDSCB algorithm utilizes the set of items $u$ has liked, $I_u^l$, in order to rank the items $u$ has not rated. When calculating a score for a given unrated item $i$, the PDSCB algorithm considers two alternative scoring approaches:

1. The first approach is a variation of the one used by the IBCF algorithm, where instead of aggregating the cosine similarity scores of all items in $I_u^l$, the PDSCB algorithm aggregates the $k$ items which had the highest cosine similarity. As will be discussed later on, $k$ is a parameter within the model. This approach is similar to the one taken by Karypis (2001). We denote this approach by “$k$ approach”.

2. The second approach starts by obtaining a single vector representation of $I_u$. This is achieved by averaging all of the $n$-dimensional vectors representing the respective items in $I_u$. The averaged vector (“the centroid”) is also an $n$-dimensional vector $\text{centroid}(I_u) = <c_1, c_2, \ldots, c_n>$, in which each entry $c_d$ represents the relevance score of the corresponding term $t_d$, averaged over the items in $I_u$:

$$c_d = \frac{1}{|I_u|} \sum_{i \in I_u} v_{id}$$
Algorithm 2: Content-based algorithm for ranking a set of unrated items, offered by service provider $s$, for a user $u$.

**Input**: $u, r_u, VS \in S, k, VS \in S, $ rating\_thresh, $k$

**Output**: Ranked list of the items $u$ has not rated

1. $l_{u, un\text{r}} ← \{i \in k | r_u(i, u) = 0\}$
2. $l_{u, rated} ← \{i \in \bigcup_{j} f_j | R_j(u, i) \neq 0\}$
3. $I_u' ← \{i \in l_{u, rated} | R_j(u, i) \geq \text{rating\_thresh}\}$
4. for item in $I_u' \cup l_{u, un\text{r}}$ do
5. $v(\text{item}) ← \text{score}_v \in \text{content}(i)) \forall d \in \{1, 2, \ldots, n\}$ end for
6. for unrated\_item in $l_{u, un\text{r}}$ do
7. all\_similarities ← \{\} $[k \text{ approach}]$
8. for liked\_item in $I_u'$ do
9. similarity ← cosine\_similarity($v(\text{unrated}\_item),$
10. \text{ ranked\_similarities.append(similarity)}$
11. end for$[\text{ centroid approach}]$
12. $k(\text{ approach\_score}) ← 0$
13. for $l$ in $\{1, 2, \ldots, k\}$ do
14. $k(\text{ approach\_score}) ← k(\text{ approach\_score}) + \text{ranked\_similarities}[l]$
15. end for
16. assign $k(\text{ approach\_score})$ to unrated\_item
17. liked\_items\_centroid ← \{0.0, 0.0, 0.0\}$[\text{ centroid approach}]$
18. for liked\_item in $I_u'$ do
19. liked\_items\_centroid ← liked\_items\_centroid + $v(\text{liked\_item})$
20. end for
21. centroid\_approach\_score ← cosine\_similarity($v(\text{unrated}\_item),$
22. \text{ liked\_items\_centroid}$[I_u']$
23. assign centroid\_approach\_score to unrated\_item
24. end for
25. return $l_{u, un\text{r}}$ ranked by either $k(\text{ approach\_score})$ or centroid\_approach\_score

Then, $i$'s assigned score is the cosine similarity between $i$'s vector and $I_u'$'s centroid. This approach is similar to the one taken by Han and Karypis (2000) and Gabrilovich and Markovitch (2007). We denote this approach by “centroid approach”.

The rationale behind examining both scoring approaches for predicting the relevance of an unrated item is to evaluate whether capturing the individual representation of items in $I_u'$ has added value over a reduced, averaged representation of $I_u'$.

4. Use cases and data

In order to evaluate the proposed method described in Section 3 above, we have used two different use cases, based on separate datasets, which have their own attributes and characteristics. The first use case is a movie recommendation system, based on the MovieLens dataset (Harper & Konstan, 2016; Vig, Sen, & Riedl, 2012). The second use case deals with a web browsing dataset and aims at recommending web pages to users.

We provide below a thorough description of each use case, including a detailed explanation of the corresponding dataset, and the manner we used it for our evaluation. In particular, we divide our description of each use case into four main components: General Description, Data Preprocessing, Data Filtering and Data Splitting.

Although these two use cases are quite specific, it is important to mention that both the architectural scheme of open-PDS/SafeAnswers and our model formulation are general and adaptive and thus could be extended to fit other use cases as well.

4.1. Use Case 1: The MovieLens dataset

4.1.1. General description

The first use case deals with the MovieLens dataset. This well-known dataset contains multiple aspects of data regarding movies, and was created and maintained by the GroupLens Research Lab at the Department of Computer Science and Engineering at the University of Minnesota. We have used two separate dataset releases:

1. MovieLens 10M Dataset (Harper & Konstan, 2016), containing 10,000,054 ratings applied to 10,681 movies by 71,567 users. In short, this dataset contains ratings, on a scale from 0.5 to 5.0, with 0.5 increments, which were applied to movies by users (both movies and users are identifiable by their ID). These ratings are explicit by definition, as the users manually assigned them to movies of their choice. In addition, this dataset also contains genres associated to movies. Each movie can be associated with multiple genres.

2. MovieLens Tag Genome Dataset (Vig et al., 2012), containing a pool of 1128 tags, and their assignment (including relevance) to 9734 movies. Tags were assigned to movies by users, a process which resulted in a finite set of possible tags, and a relevance score for each movie-tag combination, on a scale between 0 and 1. Tags are supposed to represent some aspect of the movie, for example violent, aliens and funny.

4.1.2. Data preprocessing

In order to have a dataset which meets the assumptions and requirements of our model, a few preliminary steps were required. Representing Movies as n-Dimensional Vectors. As mentioned in 3.2 above, our model relies on the representation of each item (movie in this case) as a vector in an n-dimensional space. Given the structure of the MovieLens dataset as described above, we used the tags as the dimensions in this space, resulting in $n = 1128$ tags/dimensions for each movie vector. As the values of each movie vector, we used the relevance score between the different tags and the movie (recall that these relevance scores were provided in the dataset).

Binarizing the Ratings of Movies. We transformed the discrete rating scale in this dataset to a binary one, using the transformation described in Section 3, by setting rating\_thresh to the median of all ratings in the dataset – we found this value to be 4.0. A sensitivity analysis test of setting different rating\_thresh values is discussed in 5.2.1.

4.1.3. Data filtering

Filtering Movies. Since our data comes from two different dataset releases, we performed a filtering process that kept only movies which appeared in both releases. This resulted in a set of 8429 movies.

Filtering Users. In order to deal only with users having a sufficient amount of data, we have retained users which have rated at least 1000 movies. This resulted in a set of 824 users. A sensitivity analysis test of this threshold is provided in Appendix A.

4.1.4. Data splitting

Since the MovieLens dataset comes as a coherent dataset, we used the inherent assignment of movies to genres, in order to split the movies to different service providers. More specifically, the MovieLens dataset includes 18 genres: Action, Adventure, Animation, Children’s, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War and Western, where each movie can be associated with multiple genres. For example, the movie Toy Story is associated with the genres Adventure,
Animation, Children's, Comedy and Fantasy. We treated each genre as if it was a separate service provider. For example, the genre Drama was considered as a separate service provider, offering all movies which were assigned the Drama tag. Clearly, in such case several service providers can offer the same movie.

In Table 2 we summarize, for each genre, the number of movies assigned to the genre and the number of ratings recorded to movies in the genre.

<table>
<thead>
<tr>
<th>Genre</th>
<th># of movies</th>
<th># of ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>1473</td>
<td>2,845,349</td>
</tr>
<tr>
<td>Adventure</td>
<td>1025</td>
<td>2,121,074</td>
</tr>
<tr>
<td>Animation</td>
<td>286</td>
<td>519,112</td>
</tr>
<tr>
<td>Children's</td>
<td>528</td>
<td>820,149</td>
</tr>
<tr>
<td>Comedy</td>
<td>3703</td>
<td>3,934,068</td>
</tr>
<tr>
<td>Crime</td>
<td>1118</td>
<td>1,474,957</td>
</tr>
<tr>
<td>Documentary</td>
<td>482</td>
<td>103,454</td>
</tr>
<tr>
<td>Drama</td>
<td>5339</td>
<td>4,344,198</td>
</tr>
<tr>
<td>Fantasy</td>
<td>543</td>
<td>1,028,482</td>
</tr>
<tr>
<td>Film-Noir</td>
<td>148</td>
<td>131,592</td>
</tr>
<tr>
<td>Horror</td>
<td>1013</td>
<td>768,225</td>
</tr>
<tr>
<td>Musical</td>
<td>436</td>
<td>481,174</td>
</tr>
<tr>
<td>Mystery</td>
<td>509</td>
<td>630,944</td>
</tr>
<tr>
<td>Romance</td>
<td>1685</td>
<td>1,901,883</td>
</tr>
<tr>
<td>Sci-Fi</td>
<td>754</td>
<td>1,490,489</td>
</tr>
<tr>
<td>Thriller</td>
<td>1706</td>
<td>2,584,435</td>
</tr>
<tr>
<td>War</td>
<td>511</td>
<td>506,063</td>
</tr>
<tr>
<td>Western</td>
<td>275</td>
<td>210,459</td>
</tr>
</tbody>
</table>

4.2. Use Case 2: The web browsing dataset

4.2.1. General description

Our second use case focused on web browsing data. Web browsing records have long been known to represent users’ areas of interest (Goel, Broder, Gabrilovich, & Pang, 2010), and thus a dataset consisting of such records will be of interest to the proposed content representation model. The dataset we have used was provided by a big toolbar company (we cannot provide the name of the company), and contains web browsing records. Each such record consists of multiple fields, including the user’s anonymized ID, IP, the client’s device, the HTTP request made by the user, the time in which the request was made, alongside a few technical details as well. A sample from the dataset is shown in Table 3. The original dataset consists of 2,042,695,796 records, from 5,488,897 distinct users, over a time period of approximately 13 months.

4.2.2. Data preprocessing

In order to use this dataset for our evaluation purposes, a few preliminary steps were required to prepare the data. Since the data was relatively large in size, in most of the steps described below, we were using “Big Data” tools such as Apache Spark, Apache Parquet and Cloudera Impala.

Domain Extraction. For each record we generated an additional field which held the domain name by extracting it from the URL (e.g. the domain extracted from the URL http://www.bbc.com/news/world-us-canada-38155141 was bbc.com).

Representing Web Pages as n-Dimensional Vectors. In our evaluation, we have focused on text-centric web pages (see further details below). In order to transform the text content in web pages into vectors in an n-dimensional space, we followed the Vector Space Model (VSM) approach. The basic and most essential step towards such a transformation is finding the terms which are present in a text, as stated in Table 2.

In order to have our model rely on state-of-the-art text analysis procedures, we have used for this purpose AlchemyAPI. This service provides, for a given text, a list of terms (“keywords”) in the terminology of AlchemyAPI or topics (“tagged concepts”), i.e. concepts which were tagged to an entry in some external database, such as DBpedia, which are apparent in the text, alongside relevance scores for each topic/term. The scores are scaled between 0 and 1, where a higher score means higher confidence that the given term is indeed relevant to the content of the page.

These two modeling approaches, provided by AlchemyAPI, served as an additional parameter in the model when dealing with this specific use case. We will refer to this parameter, denoted by content_space, while evaluating our model.

In the context of our work, AlchemyAPI’s analysis of each given URL includes extracting the actual text from the web page, and running natural language processing (NLP) on it. AlchemyAPI state-of-the-art text analysis algorithms were proven to perform well on various types of data, as shown in numerous studies (Adamopoulos, 2013; Cancanari, Di Iorio, Nuzzolese, Peroni, & Vitali, 2013; Kononenko, Dietrich, Sharma, & Holmes, 2012; Meehan, Lunney, Curran, & McCaughhey, 2013; Saif, He, & Alani, 2012; Saritha & Devshirioy, 2013).

For illustration, in Tables 4–5 we show the results which were retrieved from AlchemyAPI when analyzing the following text:

“Decision tree learning uses a decision tree as a predictive model which maps observations about an item (represented in the branches) to conclusions about the item’s target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a finite set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees”.

AlchemyAPI’s results can be easily plugged into the content-based vector-representation approach. Assuming an exhaustive set of terms from all pages, denoted by T (where |T| = n), each term’s relevance score, returned by AlchemyAPI, will appear in its respective dimension in the vector representation of the page.

Labeling the Vectors. It is important to note that the users’ feedbacks in this dataset are implicit, since the only information we have is whether a given user has visited a certain web page or not. Since we do not have any other data about the user-page interaction (e.g. time spent on each page, whether the user liked the page or not), we assume that visiting a page expresses the user’s interest in the given page. Moreover, if a user did not visit a certain web page, it probably does not mean she has a negative sentiment for that page. However, for the purpose of this work, we regard such cases as a lack of a positive sentiment. In the terminology used in Section 3, applied to each page, if a given user has visited a certain page, it means she liked it, and she did not like the page otherwise.

---

2. In fact, AlchemyAPI handles URLs as well. The system then extracts the text out of the web page. This is the way that we have used the API throughout this work.
5. As will be described later on, we have implemented our model both for topics (“concepts”) and for terms (“keywords”) space models. For the sake of convenience, while describing the model generally, we will refer only to the terms space model, as we have done up to this point.
6. This text is the first paragraph of http://en.wikipedia.org/wiki/Decision_tree_learning.
Table 3
Web browsing data sample.

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Source ID</th>
<th>User ID</th>
<th>IP</th>
<th>Country ID</th>
<th>Client agent</th>
<th>Previous site</th>
<th>Requested site</th>
</tr>
</thead>
</table>

Table 4
AlchemyAPI results - keywords.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Confidence score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictive Modeling Approaches</td>
<td>0.90704</td>
</tr>
<tr>
<td>Target Variable</td>
<td>0.88338</td>
</tr>
<tr>
<td>Tree Learning Uses</td>
<td>0.83637</td>
</tr>
<tr>
<td>Represent Class Labels</td>
<td>0.77890</td>
</tr>
<tr>
<td>Target Value</td>
<td>0.64357</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>0.62343</td>
</tr>
<tr>
<td>Decision Trees</td>
<td>0.54179</td>
</tr>
<tr>
<td>Continuous Values</td>
<td>0.50385</td>
</tr>
<tr>
<td>Finite Set</td>
<td>0.47798</td>
</tr>
<tr>
<td>Data Mining</td>
<td>0.46993</td>
</tr>
<tr>
<td>Regression Trees</td>
<td>0.46227</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>0.45517</td>
</tr>
<tr>
<td>Real Numbers</td>
<td>0.44822</td>
</tr>
<tr>
<td>Tree Models</td>
<td>0.44337</td>
</tr>
<tr>
<td>Classification Trees</td>
<td>0.43762</td>
</tr>
<tr>
<td>Tree Structures</td>
<td>0.43653</td>
</tr>
<tr>
<td>Item</td>
<td>0.30266</td>
</tr>
<tr>
<td>Branches</td>
<td>0.26814</td>
</tr>
<tr>
<td>Leaves</td>
<td>0.23370</td>
</tr>
<tr>
<td>Conjunctions</td>
<td>0.22639</td>
</tr>
<tr>
<td>Conclusions</td>
<td>0.21533</td>
</tr>
</tbody>
</table>

Table 5
AlchemyAPI results - concepts.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Confidence score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>0.90704</td>
</tr>
<tr>
<td>Tree</td>
<td>0.87143</td>
</tr>
<tr>
<td>Decision Trees</td>
<td>0.856374</td>
</tr>
<tr>
<td>Decision Analysis</td>
<td>0.76849</td>
</tr>
<tr>
<td>Influence Diagram</td>
<td>0.73624</td>
</tr>
<tr>
<td>Set Theory</td>
<td>0.73001</td>
</tr>
</tbody>
</table>

4.2.3. Data filtering

**Content Type Filtering.** As described above, our user modeling process is heavily reliant on text analysis, a fact which led us to filter out requests for web pages containing non-text-centric content, e.g., videos and photos. This was done by creating domain-specific rules regarding its URL structure.

**Language and Location Filtering.** In order to eliminate the possibility of our model's evaluation being biased due to language-specific or geographic location effects, we have retained only the records of users from the USA. This was possible using the "country code" field in the original dataset.

**Users and Domains Filtering.** As previously mentioned, one of the goals of this work is to check whether the combination of data from multiple service providers will allow better modeling of the users. Our original plan with this dataset was to treat different domains as different service providers. Unfortunately, the required preprocessing and filtering actions which were made on the original dataset resulted in a very small number of users who were active across different domains.

To overcome this problem, we decided to focus on the domain with the largest number of records and divide it synthetically into different service providers as will be described in detail in the next subsection.

Out of all domains, Wikipedia.org had the largest number of records, with a total of 848,009 records made by 51,273 users, who have visited 286,865 distinct pages in this domain. In order to guarantee that each user has a sufficient amount of data (even after dividing the dataset into the different service providers), we retained only users who have visited at least 100 distinct pages in Wikipedia.org. This resulted in a set of 376 users, who have visited 61,084 distinct pages, in a total of 157,899 records.

It is important to note that the evaluation had an extremely long runtime, and therefore we were not able to perform a comprehensive examination of this threshold (i.e., a sensitivity analysis test for this threshold was not performed).

4.2.4. Data splitting

In order to synthetically divide Wikipedia.org into different service providers, we simply randomly split its set of URLs to mutually exclusive and collectively exhaustive subsets of URLs. The rationale behind splitting the URLs randomly was to allow an overlap between the content of the different subsets (i.e., pages from different subsets dealing with similar topics), while at the same time enabling some distinction between the subsets (i.e., a page dealing with a specific unique subject will appear in a certain subset and not in the others).

As will be discussed in the evaluation of this use case, we have used a parameter, denoted as division_factor, which determines to how many subsets we divide the URLs set.

5. Evaluation

After thoroughly describing the proposed method, including a detailed description of our model, and the specific use cases (and datasets) we have used for the purpose of this work, our evaluation is focused on three main aspects:

1. Test whether the proposed privacy-enhancing model (PDSCB) can achieve similar (or better) recommendation accuracy in comparison to the existing alternative (IBCF).
2. Test whether the recommendation accuracy of PDSCB can be improved when exploiting data of multiple service providers (as opposed to a single source of data).
3. Examine the impact of various features and parameters within the proposed PDSCB model.

It is important to note that our evaluation focuses on the accuracy of the recommendation system, while assuming that the reliance on the openPDS architecture inherently offers an enhanced level of privacy to the user.

We begin by describing the evaluation methodology we have used, followed by the obtained results.

5.1. Evaluation methodology

As detailed in Section 3, we assume a set of users $U$ and a set of service providers $S$. Each $s \in S$ offers a set of items $I_s$ and a rating matrix, denoted as $R_s$. In addition, recall from 3.2 that we assume
T, an exhaustive set of terms from all items in the dataset. As discussed, |T| = n, which enables the representation of each item as an n-dimensional vector.

The methodology we have used in order to evaluate our model can be described by the following steps. For the sake of convenience, the evaluation process below is described for a given user u, and a specific service provider s. The evaluation was made after repeating the following steps for all users and all service providers in the dataset.

The evaluation of the IBCF model was implemented using the LibRec library, with default parameters values (Guo, Zhang, Sung, & Yorke-Smith, 2015).

5.1.1. Data split to training and test sets

Given a user u ∈ U, we denote the set of items she has liked in s as $I_{su}$. Note that in Section 3, where we did not specify the service provider in which u has shown interest, we denoted this subset of items by $I_u$. We divide $I_{su}$ to a training set (70%), denoted by $I_{su, train}$, and a test set (30%), denoted by $I_{su, test}$. As a general convention, this split was done randomly. As described below, we make extensive use of the set of items in the union of the training sets from all users. We denote this united train set as $I_{train, union}$.

5.1.2. Ranking unrated items

As described in Section 3, for a given user u, we treat $I_{su}$ as the set of items u has liked. That is, we use this set of items in order to provide a score for an unrated item.

The subset of items upon which we predict u’s level of interest is $I_{train, union} \setminus I_{su, train}$. We apply to these items the prediction procedure, described in Section 3. From this process we obtain for u, for each i ∈ $I_{train, union} \setminus I_{su, train}$, a score, which will then be used to rank all items. As stated in Section 3, our model focuses on the ranking of items, rather than on predicting their rating.

5.1.3. Labeling unrated items

As described in the previous subsection, the model assigns for u a score for each i in $I_{train, union} \setminus I_{su, train}$. In order to enable the evaluation approach of a supervised classification problem, we labeled each of these items as either ‘1’ or ‘0’. The labeling was made according to the presence of an item in $I_{su, test}$: if a given item i ∈ $I_{train, union} \setminus I_{su, train}$ is present in $I_{su, test}$, its assigned label will be ‘1’, otherwise the label will be ‘0’.

One might suggest applying the prediction procedures described above to the set of items $I_{su, test}$ instead of $I_{train, union} \setminus I_{su, train}$. Since the former contains both ‘1’ and ‘0’ labeled items, and thus suits better an evaluation procedure of a classification algorithm via a test holdout set. The reason for our selection in the latter option resides in the fact that this procedure is more generalizable, since it supports also cases in which the user’s data contains only positive (i.e. ‘1’ labeled) items, so the user’s feedback for negative items is more implicit. In fact, the web browsing dataset is an example for such a use case (see Section 4.2).

5.1.4. Using additional service providers’ data

As described above, u’s training set from a service provider s, denoted as $I_{su}$, is the set of items the model considers as items u has liked, when providing scores for other items offered by s. In order to evaluate our hypothesis regarding the enhancement of the recommendation process when adding service providers’ data in the PDSCB model, we repeat the scoring procedure twice - once where the training set contains u’s training set only from s (i.e. $I_{su, train}$), and another time where the training set includes $igcup_{j \in S} I_{su, train}$. Both prediction iterations were applied to the same set of items, $I_{train, union} \setminus I_{su, train}$.

This enables us to later evaluate whether the addition of u’s data, originating from service providers other than s, will improve the recommended items’ fit to her interests.

5.1.5. Evaluation metrics

As referred to earlier, the output of our model, for each user, is a ranked list where each item in the list is an unrated item, and a score is assigned to each item, where a higher score means a higher level of interest in the given item. In addition, all items upon which the prediction is applied are labeled as part of the evaluation phase. That is, items the user has liked are labeled as ‘1’, and the rest of the items as ‘0’. In our evaluation we used the area under the ROC curve (AUC) metric, which is commonly used for evaluating ranking algorithms (Herlocker, Konstan, Terveen, & Riedl, 2004; Tiroshi & Kuflik, 2012). In addition, we use the precision at n and recall at n metrics, which are widely used in use cases similar to the ones examined in this work (Schütze, 2008).

5.1.6. Examined parameters

As stated above, we are interested in inspecting various parameters within the PDSCB model. Here we describe the parameters which were mutual to both use cases mentioned in Section 4. Some other aspects were relevant only to a specific use case, and thus will be discussed in a use case specific manner, later on in this section.

1. Relevance Score Threshold. As described above, each term is assigned a relevance score, typically in the range [0,1]. In our model we define score_threshold, a threshold used to discard terms with scores lower than it. score_threshold in fact controls the level conservatism of the model - a low threshold will accept most of the results, whereas a higher one will accept only results which were returned with high confidence. We were interested in inspecting the effect of the threshold set for these scores, across the range of possible threshold values.

2. Scoring Approach. As mentioned above, the estimation of a given unrated item’s relevance for a user can be calculated using two methods: (1) k approach - by measuring its similarity to each individual item liked by the user and then aggregating the k (see below) highest ranked items, or (2) centroid approach - by measuring its similarity to the centroid of items liked by the user. We examine the effect the scoring approach has on the results.

3. k’s Value. k’s value varies as well within the model, where smaller values imply measuring the unrated item’s relevance according to the most similar items previously liked by the user, and higher values tend to account for the averaged similarity among these items.

5.2. Results

5.2.1. Use Case 1: The MovieLens dataset

Since our evaluation for this use case involves four different free parameters, in order to reduce runtime and simplify the reporting of the results, each parameter is assigned with a default value as follows (in most cases, this value represents the median of the examined range):

- score_threshold is set to 0.5.
- The scoring approach for predicting the relevance of an unrated item to a given user is the k approach, with k = 5.
- rating_threshold is set to 4.0, which is the median of ratings in the dataset.
• *idf transformation* was not used. We will relate to this parameter later in this section.

Then, when inspecting the effect of a certain parameter (i.e., in order to test the sensitivity of the model to that parameter), as done in Figs. 6–9, the rest of the parameters were fixed to their default values.

As stated above, the main goal of our evaluation is to check whether the proposed privacy-enhancing model (PDSCB) does not suffer from a significant downgrade in recommendation accuracy, in comparison to the common alternative (IBCF). Fig. 3 shows a comparison between the existing IBCF model and two variants of the suggested PDSCB model - one that uses data only from one genre (imitating a single service provider) and one that uses data from all genres (imitating the combination of data from all service providers). As can be seen, IBCF (AUC=0.9, STD=0.01) performs much better than PDSCB which relies solely on one genre's data (AUC=0.78, STD=0.03). However, when exploiting the user's data from other genres as well, the PDSCB approach manages to outperform the IBCF approach (AUC=0.92, STD=0.04). This latter result was found to be statistically significant using a paired *t*-test with *p*-value=0.024. These results imply that exploiting the data-sharing ability, offered by the PDS architecture, can enable the proposed privacy-enhancing model to perform at least as good as the common collaborative filtering approach alternative.

The above results align with the ones shown in Figs. 11 and 12, presenting the PDSCB model's precision and recall at *n*, respectively.

The rest of this section is dedicated to inspect the impact of different parameters on the PDSCB model.

In Fig. 6, we inspect the effect of the *score_thresh* parameter, which sets a relevance score threshold for considering tags in the vector representing the movie. We do so both for recommendations based solely on one genre and for recommendations enhanced by data from all genres. Generally speaking, increasing the threshold might be desirable for (1) reducing the dimensionality of the tags space, and thus gaining improvement in terms of storage and performance, and (2) having a stronger confidence in the assigned tags. However, in this case we see that these advantages might come in the expense of a drop in the model's performance, which occurs when *score_thresh* is increased. In addition, we see that the gap between the two configurations seems to be robust and hardly changes as a function of the threshold.

In Fig. 7, we show the results of the PDSCB model for the different scoring approaches (*centroid* vs. *k*). As can be seen, when all genres' data are used, *k* approach outperforms centroid approach, whereas when only one genre's data are used there is hardly any difference in performance between the two approaches. Interestingly, when *k* increases, the performance of the *k* approach which is based on all genres' data decreases, while we don't see such an effect for the centroid approach that is based on a single genre's data.

Inverse Document Frequency (IDF) is commonly used in VSM to correct for terms that appear frequently across the different items. We wanted to test whether using IDF transformation in our use case - i.e. assigning an increased weight to tags which appear
only in few movies, and thus are assumed to be more informative - can yield better recommendation accuracy. More formally, the IDF transformation involved multiplying the relevance score of each tag with the following factor:

$$\text{idf}(\text{tag}_i) = \log \left( \frac{\text{# of movies in the data}}{\text{# of movies with tag}_i + 1} \right)$$

Fig. 8 presents a comparison between four variants of the PDSCB model - with and without IDF transformation, and using all genres’ data vs. a single genre’s data. The results suggest that the transformation does not improve the model’s performance.

As described in 4.1, in this dataset the users’ ratings were passed on a discrete, non-binary scale. In Fig. 9, we present the PDSCB model results as a function of rating_thresh, the threshold used for binarizing the users’ ratings. We do so both for recommendations based solely on one genre and for recommendations enhanced by data from all genres. In both configurations, there seems to be an improvement in the model’s performance as the threshold is increased. As can be seen, the difference between the two configurations seems to be robust and hardly changes as a function of the threshold.

5.2.2. Use Case 2: The web browsing dataset

Similar to use case 1, since our evaluation for this use case involves four different free parameters, in order to reduce runtime and simplify the reporting of the results, each parameter is assigned with a default value as follows (in most cases, this value represents the median of the examined range):

- score_thresh is set to 0.5.
- The scoring approach for predicting the relevance of an unrated item to a given user is the centroid approach.
- division_factor is set to 5.
- content_space is set to concepts.

Then, when inspecting the effect of a certain parameter (i.e., in order to test the sensitivity of the model to that parameter), as done in Figs. 10, 13, 14 and 15, the rest of the parameters were fixed to their default values.

Fig. 10 shows a comparison between the IBCF model and two variants of the PDSCB model - one that uses only one subset’s data and one that uses all subsets’ data - for different values of division_factor. As can be seen, IBCF (AUC=0.54) performs much worse than PDSCB (AUC values range between 0.59 and 0.73), even when the latter uses only 10% of the available data. These results are suggesting that in this setting, PDSCB not only enhances users’ privacy but also significantly increases the accuracy of the recommendation system. Additionally, as expected, the PDSCB model performance worsen as division_factor increases.

The above results align with the ones shown in Figs. 11 and 12, presenting the PDSCB model precision and recall at n, respectively. The rest of this section is dedicated to inspect the impact of different parameters on the PDSCB model.

In Fig. 13, we inspect the effect of score_thresh. We do so both for recommendations based solely on a subset of the data and
for recommendations enhanced by data from all other subsets. While the potential advantages of increasing the threshold were discussed in the scope of the previous use case, it is demonstrated here again that these benefits might be shadowed by degraded performance of the model. In addition, also in this use case the gap between the two configurations seems to be robust and hardly changes as a function of the threshold.

In Fig. 14, we show the results of the PDSCB model for the different scoring approaches (centroid vs. k). As shown in the figure, the difference between using the two approaches is negligible both when using one subset of the data and when using all subsets.

Interestingly, in comparison to the previous use case’s results, we don’t see a significant decrease in performance when k increases. The reason for this might be stemmed in what is often referred to as the curse of dimensionality. As extensively discussed throughout this work, our model relies on the vector representation of items in an n-dimensional space. Comparing these spaces in both use cases, in the MovieLens use case the vector space consisted of n = 1128 dimensions (i.e. tags), whereas in this use case’s configuration n = 85,407. In such high dimensional spaces, the items are typically very far from each other, and thus the notion of neighborhood or closeness between items becomes somewhat problematic. We have used the cosine similarity measure between vector-represented items, which is less exposed to this problem, since it cares for the direction of vectors rather than their length. However, these results might imply that even this calculation method is degraded when the dimensionality is so high, since exploiting the similarity between individual items cannot improve the model significantly.

In Fig. 15, we compare the PDSCB model results for both content_space options (concepts and keywords). Here as well, we do so both for recommendations based solely on a subset of the data and for recommendations enhanced by data from all other subsets. As can be seen, the concepts space performs better when exploiting data from all subsets, whereas the keywords space does better when using one subset’s data. The explanation for this might also have to do with the curse of dimensionality. As demonstrated in
Tables 4 and 5, AlchemyAPI returns, for a given text, many more keywords than concepts. For example, when comparing the created spaces for data from all subsets, the concepts space consists of $n = 85, 407$ dimensions, whereas in the keywords space $n = 1, 091, 607$. It might be the case that for large datasets, the keywords space becomes too sparse and thus hurts the performance of the PDSCB model.

### 6. Conclusions and discussion

This work has suggested an architecture for a PDS-based recommender system, which aims at enhancing the users’ privacy while not sacrificing the recommendation accuracy. In Section 3 we have provided a detailed description of the suggested PDS content-based model (PDSCB), and its common alternative, i.e. the item-based collaborative filtering (IBCF) model. In Section 5, using the two use cases described in Section 4, we have evaluated the PDSCB model, including a thorough inspection of its parameters.

Our evaluation focused on (1) comparing the suggested privacy enhancing model to its well established and popular alternative, and (2) examine the potential improvement obtained by exploiting data from multiple service providers, as enabled by the openPDS architecture, rather than a single data source. The results suggest that the proposed privacy enhancing model is able to perform as well as or better than the common collaborative filtering alternative. In addition, the use of data from multiple service providers has been shown to improve the accuracy of the recommendations. These findings are promising in the sense of offering users an architecture which is both privacy preserving and novel in the way it is able to improve its performance using additional service providers’ data.

One limitation of this work has to do with the datasets which were used in order to evaluate our model. In both use cases, the dataset did not contain separate sources of data which represented separate service providers. Thus, in order to use these datasets, we have synthetically imitated a setting of separate service providers. Whereas in Use Case 1 (MovieLens) the split of the dataset was done according to some attribute in the dataset, in Use Case 2 (web browsing data), we have randomly split the Wikipedia.org URLs set, which is obviously quite far than a setting of a dataset consisting of multiple, separate data sources. Thus, the recommendations in this work fall, in terms of the formulation presented by Cantador, Fernández-Tobías, Berkovsky, and Cremonesi (2015), into the category of attribute-level cross-domain recommendations (e.g. using genre A’s data in order to recommend movies from genre B). However, a more intriguing examination would inspect item-level cross-domain recommendations, which exploit data from different types (e.g. using books purchasing history in order to recommend movies). The general approach of item-level cross-domain recommendations has been proven to be successful in a variety of studies (Shapira, Rokach, & Freilikhman, 2013; Winoto & Tang, 2008), but the available datasets which were used in the scope of this work did not allow its inspection in the context of our PDS-inspired content-based model.

Another limitation arises from the privacy perspective. Since the suggested architecture in this work relies on the openPDS architecture, it suffers from the same issues and shortcomings, e.g. finding the right level of data aggregation for preserving privacy, attacks by malicious applications, protection of proprietary intellectual property and hosts attacks (de Montjoye et al., 2014).

Finally, the PDSCB model suffers a weakness which is mutual to all content-based recommender systems, namely its dependency on content. That is, as opposed to memory-based models, which can exploit the basic user-item matrix, content-based models need content which is the basis for their modeling and item-representation. In the case of absence or change in the content, the model would have to be rebuilt.

There are several directions in which the research which was done in the scope of this work can be expanded in future work:

- Use different datasets with different attributes for evaluating the proposed model. Moreover, it is desirable to compare the proposed model to additional state-of-the-art recommendation algorithms.
- Combine the proposed content-based model with a context-based one, using data suitable for context-aware recommendations (e.g. location and accelerometer data). The exploitation of such data is possible in the architecture of openPDS (de Montjoye et al., 2014).
- The model which we have used to rank unrated items could be extended to predict the ratings of the items, using regression models.
- After binarizing the items to ones the user had liked and others she has not liked, our content-based approach used the similarity measure between liked items and unrated ones. Another approach, which was outside the scope of our work, would tackle the ranking task by using machine learning classification algorithms (e.g. SVM) in order to generate a score for each individual item. In addition, different approaches could have been used in order to model the content, creating different vector spaces.
- We have shown that in both use cases we have evaluated, our proposed content-based algorithm outperforms the item-based collaborative filtering one. Another intriguing question is how a collaborative filtering approach would have performed in the openPDS architecture (i.e. exploiting data from multiple service providers, in a secured manner).

### Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

### Appendix A. Sensitivity analysis of the requested number of ratings per user

Recall that in our evaluation of Use Case 1 we requested that each evaluated user had rated at least 1000 movies. However, what is the impact of this threshold on the results we obtained? To emphasize the impact of this threshold further, we note that the default configuration we have used requested 1000 rated movies per user, resulting in 824 users. However, requesting 2000 movies per user resulted in only 91 users, and requesting 200 movies per user resulted in 13,651 users. In this section, we perform a sensitivity analysis of this threshold.

Fig. A.1 reports the results for the PDSCB model for different values of the requested number of rated movies per user. In addition, we examined two variants of PDSCB - one that is based solely on one genre’s data and another that is based on all genres’ data. As can be seen, in both configurations, there seems to be a decrease in the model’s performance as the threshold is increased. This might be explained by the fact that as the threshold increases, the number of movies per user drops dramatically, resulting in less significant representation of each user’s preferences. However, as clearly seen in the figure, the gap between the two variants of PDSCB seems to be robust and hardly changes as a function of the threshold.

It is also important to note that this threshold had a strong effect on the runtime of our evaluation, where lower threshold
values resulted in significantly longer runtimes. After inspecting the results shown in this appendix, we chose 1000 as the threshold value, leveraging both a significant amount of movies rated by each user and having hundreds of such users at the same time.

Credit authorship contribution statement

Itzik Mazeh: Data curation, Formal analysis, Investigation, Methodology, Visualization, Writing - original draft, Writing - review & editing. Erze Shmueli: Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing.

References


