IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 24, 2018, accepted June 15, 2018, date of publication June 26, 2018, date of current version July 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2850762

Lightweight Tag-Aware Personalized Recommendation on the Social Web Using Ontological Similarity

ZHENGHUA XU[1], OANA TIFREA-MARCIUSKA[2], THOMAS LUKASIEWICZ[1], MARIA VANINA MARTINEZ[3,4], GERARDO I. SIMARI[3,4], AND CHENG CHEN[5]

[1]Department of Computer Science, University of Oxford, Oxford OX1 3QD, U.K.
[2]Bloomberg, London EC4N 4TQ, U.K.
[3]Departamento de Ciencias e Ingenieria de la Computacion, Universidad Nacional del Sur (UNS), (8000) Bahia Blanca, Argentina
[4]Instituto de Ciencias e Ingenieria de la Computacion (CONICET-UNS), (8000) Bahia Blanca, Argentina
[5]China Academy of Electronics and Information Technology, Beijing 100041, China

Corresponding author: Zhenghua Xu (zhenghua.xu@cs.ox.ac.uk)

This work was supported in part by The Alan Turing Institute under the UK EPSRC Grant EP/N510129/1, in part by an EPSRC Doctoral Prize under the EPSRC Grant EP/N509711/1, in part by the EPSRC Grants EP/J008346/1, EP/R013667/1, EP/L012138/1, and EP/M025268/1, in part by a Google European Doctoral Fellowship, in part by a Yahoo Research Fellowship, in part by the Universidad Nacional del Sur and CONICET, Argentina, in part by the EU H2020 Research And Innovation Program for the project "MIREL" under the Marie Sklodowska-Curie Grant 690974, and in part by the U.S. Department of the Navy, Office of Naval Research, under Grant N00014-15-1-2742.

ABSTRACT With the rapid growth of social tagging systems, many research efforts are being put into personalized search and recommendation using social tags (i.e., folksonomies). As users can freely choose their own vocabulary, social tags can be very ambiguous (for instance, due to the use of homonyms or synonyms). Machine learning techniques (such as clustering and deep neural networks) are usually applied to overcome this tag ambiguity problem. However, the machine-learning-based solutions always need very powerful computing facilities to train recommendation models from a large amount of data, so they are inappropriate to be used in lightweight recommender systems. In this paper, we propose an ontological similarity to tackle the tag ambiguity problem without the need of model training by using contextual information. The novelty of this ontological similarity is that it first leverages external domain ontologies to disambiguate tag information, and then semantically quantifies the relevance between user and item profiles according to the semantic similarity of the matching concepts of tags in the respective profiles. Our experiments show that the proposed ontological similarity is semantically more accurate than the state-of-the-art similarity metrics, and can thus be applied to improve the performance of content-based tag-aware personalized recommendation on the social web. Consequently, as a model-training-free solution, ontological similarity is a good disambiguation choice for lightweight recommender systems and a complement to machine-learning-based recommendation solutions.

INDEX TERMS Folksonomies, ontological similarity, personalized recommendation, social tags.

I. INTRODUCTION

In recent years, there has been an exponential increase in the popularity of the Social Web (Web 3.0), a term frequently used to describe a collection of social platforms that link people through the (World Wide) Web. The Social Web is now the basis of many online activities: shopping (e.g., eBay and Amazon), entertainment (e.g., YouTube and Last.fm), and social networking (e.g., Facebook and Twitter). With the continuing rapid growth of the Social Web, users usually can no longer browse efficiently through all the information available online due to information overload. Thus, personalized recommender systems have arisen on the Social Web, which automatically filter out irrelevant online information and provide personalized user recommendations. How to provide accurate recommendation has been widely studied in the research community [1]–[4], and commercial applications have been widely deployed; concrete examples include music recommendation on Last.fm, item recommendation on Amazon.com, and friend recommendation on Twitter.

A. MOTIVATION

To realize personalized recommendations, recommender systems are typically built based on leveraging the users' personal interests or preferences to build user profiles. The profiles are subsequently used for personalized re-ranking, where items in a database are re-ranked according to the given user profile such that items reflecting a user's personal interests are ranked higher [5]. In general, the users' preferences or interests can be obtained by leveraging information that is either *explicitly* or *implicitly* available.

Recommender systems that use explicit preference information usually require extra efforts from users to fill in preference information or for users to provide positive or negative feedback for the given recommendation results [6]. The quality of the resulting user profile is therefore strongly dependent upon the user's willingness and capability to provide sufficient and appropriate preference information. As a result, recommendation performance can be very unstable.

As an ''effortless'' alternative, implicit user activity data such as the query history [7]–[10], the browsing history [11], the user's current tasks [12] or intents [13], and even eye-tracking during search sessions [14] have been used as supplementary information about users in the recommender systems of modern websites and applications such as Google, Amazon, Youtube, and Facebook. However, due to privacy, ethical, legal, or commercial reasons, these data are usually not accessible by external parties. By accepting a ''user agreement'', users may allow the service provider to collect and use their implicit activity data to develop a better user experience, but it is normally prohibited to then share these data with third parties, and large IT companies are generally reluctant to release these valuable data to other parties.

Therefore, the need for publicly available and easy accessible user data is compelling for the development of personalized recommender systems on the Social Web. Fortunately, users can now freely provide *social annotations* to online items (e.g., Web pages, songs, videos, or other online resources) on the Social Web via bookmarking, tagging, rating, or commenting. In fact, such social annotations are actually ideal user data for personalized recommendation on the Social Web for the following reasons [15], [16]:

- Social annotations are usually publicly available online, so they are generally easily accessible (with the permission of users and/or website owners).
- Social annotations are provided by users directly as their individual opinions about online items, so the interests and preferences of users can be harvested from their social annotations.
- The aggregation of tags assigned to an item can be seen as the social summary of this item, which is helpful for recommending items with little textual content.

B. SOCIAL TAGGING PROBLEMS OVERVIEW

A *social tagging system* is an online system on the Social Web that enables users to create tags to annotate and categorize Social Web resources such as Web pages, songs, and videos.

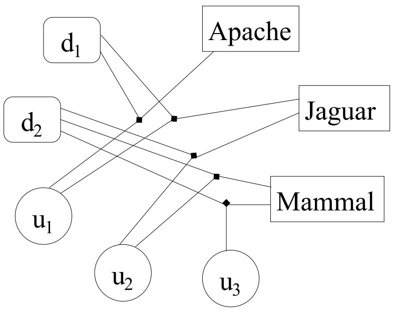

FIGURE 1. Example of a folksonomy.

This practice is called *social tagging* or *collaborative tagging*, while the classification systems derived from social tagging activities are known as *folksonomies*. An example of a folksonomy resulting from the tagging activities in a social tagging system is shown in Figure 1, where (i) user u_1 uses two tags, *Apache* and *Jaguar*, to annotate the document d_1, (ii) user u_2 uses two tags, *Jaguar* and *Mammal*, to annotate d_2, and (iii) user u_3 uses a tag, *Mammal*, to annotate d_2.

As a typical social annotation, social tags have been widely used for tag-aware personalized recommendation on the social Web based on *content-based filtering* [15], [17]–[21] or *collaborative filtering* [22], [23]. To this end, content-based filtering is especially important and attracts more and more attention in the research community [15], [17], [18], [20], [21] because it allows recommender systems to incorporate tags as additional content information to improve recommendations [24]. In this paper, we focus on personalization using social tags, but our techniques can easily be adapted to other social annotations, such as comments and blog posts.

As for content-based tag-aware personalized recommendation, to achieve personalization, a similarity measure is required to estimate the relevance of a user's preferences (described by a *user profile*) and the social summary of documents (described by a *document profile*), where the vector space model (VSM) [25] is used to represent both profiles as weighted vectors of tags. Precisely, a user's profile is usually obtained by aggregating all the social tags assigned by this user to the online documents; it is represented as a weighted vector of tags, where each dimension corresponds to a tag applied by this user, and the value of each dimension is the weight of the corresponding tag, which is influenced by the number of times that the tag is applied by the user in his/her bookmarking activities. For example, the user profile of the user u_1 in Figure 1 is $\vec{p}_{u_1} = \{(Apache, 1), (Jaguar, 1)\}$. Similarly, the document profile is obtained by aggregating all the social tags assigned by users to this document; it is also represented as a weighted vector, where the weight of each dimension is influenced by the number of times that the document is bookmarked with the corresponding tag. For example, the document profile of d_2 in Figure 1 is $\vec{p}_{d_2} = \{(Jaguar, 1), (Mammal, 2)\}$.

To calculate the similarity between the weighted vectors of these two profiles, the most widely adopted measure is *cosine similarity* [15], [22]. In addition, Vallet *et al.* [20] propose *scalar similarity* to try to improve the metric by eliminating the vector length normalization factors in cosine similarity—the rationale behind this modification is the belief that the presence of a large number of related tags is correlated with the popularity of the documents among users, and normalization thus penalizes popular documents.

However, these two state-of-the-art similarity metrics can only match tags *literally*, and use their semantic content only to a very limited extent. Since users can freely choose their own vocabulary of tags (without specifying their relationships), the resulting tags associated with documents usually contain much ambiguous information, e.g., as homonyms (tags with the same spelling but with different meaning) or synonyms (different tags with the same meaning). Thus, performing a literal matching over such ambiguous tags may lead to inaccurate similarity values between corresponding profiles, which greatly degrades the performance of the content-based recommendation system. For example, in Figure 1, two users both use the tag "Jaguar", but the two tags actually have a different meaning: u_1 uses "Jaguar" as a type of military aircraft, while u_2 uses "Jaguar" to refer to a large cat. Although these two concepts are semantically different, the cosine and the scalar similarity of two "Jaguar" tags are always 1 (identical). In contrast, "College" and "University" are similar in semantics, but their cosine and scalar similarity are 0 (irrelevant).

One existing solution to the tag ambiguity problem is to disambiguate synonyms and homonyms using their most related tags, which are the tags having the highest relatedness to the given tag, measured by the similarity between their vector space representations, e.g., tag context, document context, and user context representations [26], [27]. However, since the social data is usually very sparse, the tag disambiguation performance of this solution is limited. Another existing solution is to first apply machine learning techniques, e.g., clustering [19] or deep neural networks [21], [28] in the tag space to model abstract feature representations for tag-based user and document profiles, and then use the resulting abstract representations to estimate the similarities between users and documents for personalized recommendation.

Although machine-learning-based solutions can overcome the tag ambiguity problem and achieve good recommendation performance on the Social Web, they share the following shortcoming. In order to achieve a good recommendation performance, machine-learning-based recommendation solutions must always be trained using a large amount of data, and this training process needs to be invoked frequently to capture the dynamic changes on the Social Web. Consequently, large user datasets and powerful computing facilities are required to support such online recommender systems, which would not be generally available for small-size start-up companies or independent developers, though they are not a problem for tech giants like Facebook or Amazon. Therefore, this may prohibit the use of machine-learning techniques in lightweight recommendation systems that have very few users.

C. OUR MAIN CONTRIBUTIONS

To alleviate the need for powerful computing facilities to train recommendation models, in this paper we leverage ontology techniques [29] and propose ontology-based personalized recommender systems that apply a novel similarity metric, called *ontological similarity*, to tackle the tag ambiguity problem without the need for model training. The novelty of this ontological similarity is that it first leverages external domain ontologies to disambiguate tag information and then semantically quantifies the relevance between user and item profiles according to the semantic similarity of the matching concepts of tags in the respective profiles. Ontological similarity is semantically more accurate than the state-of-the-art similarity metrics, and can thus be applied to improve the performance of content-based tag-aware personalized recommendation on the Social Web. In addition, as a model-training-free solution, ontological similarity is a good disambiguation choice for lightweight recommender systems and a complement to machine-learning-based recommendation solutions.

More specifically, in this work we propose a two-step top-down disambiguation algorithm to address the tag ambiguity problem by mapping tags to unique matching concepts in the ontology: given either a user or an item profile, we first map tags in its profile to all the possible concepts in the ontology (this is called the *mapping step*); however, due to homonyms, it may be possible to map a certain tag to multiple concepts (this is called *multiple occurrence*); e.g., "Jaguar" may refer to a large cat, a brand of car, a military aircraft, or a tank destroyer, so it can be mapped to four concepts in different contexts. To overcome this challenge, we then propose a top-down traversal solution to use the statistical information of matching concepts of other tags in the same profile as context to disambiguate tags with more than one matching concept (this is called the *disambiguation step*).

After tag disambiguation, each tag will have at most one matching concept in the ontology, which enables us to use the semantic relevance of concepts in ontologies, called *concept similarity*, to estimate the semantic similarity of the corresponding tags. Consequently, the *ontological similarity* is computed as the weighted sum of all individual concept similarity values, which is an integration of the concept similarity and the classic cosine or scalar similarity. Intuitively, this proposed ontological similarity will be able to tackle the tag ambiguity problem by mapping homonyms (resp., synonyms) to semantically very different (resp., close) concepts in the ontology based on the different (resp., similar) profile context, resulting in a low (resp., high) ontological similarity between homonyms (resp., synonyms).

In summary, this paper makes the following contributions:

- To alleviate the need for powerful computing facilities to train recommendation models, we propose *ontological similarity* to circumvent the tag ambiguity problem for applications in lightweight recommender systems on the Social Web.

- A two-step top-down disambiguation algorithm is first developed to solve the tag ambiguity problem by mapping tags to unique matching concepts in the ontology. An algorithm is then proposed to use the concept similarities in the ontology to compute the ontological similarities between profiles. Finally, an algorithm that uses ontological similarity for content-based tag-aware personalized recommendation is proposed. The computational complexity of these algorithms is also investigated.
- We perform extensive experiments based on a public *Delicious* dataset and evaluate the performance of the proposed algorithms from three perspectives: content-based personalized recommendation, tag disambiguation, and tag-to-ontology mapping. The experimental results show that:
  - (i) The proposed ontological-similarity-based recommender systems are more effective than the state-of-the-art cosine-similarity-based and scalar-similarity-based recommender systems in content-based tag-aware personalized recommendations in terms of *all* evaluation metrics.
  - (ii) The recommendation accuracy of our ontological-similarity-based solution is better than those of the clustering and autoencoder baselines, but is worse than that of the DSPR baseline; however, its computational cost is much lower than those of all machine-learning-based approaches (especially, it is 118.4 times quicker than DSPR). Therefore, due to its low computational cost and reasonable recommendation accuracy, the proposed ontology-based solution is a good disambiguation choice for lightweight recommender systems and a complement to machine-learning-based recommendation.
  - (iii) The proposed ontology-based solution greatly outperforms (with more than double the performance) the state-of-the art baselines in tag disambiguation.
  - (iv) The proposed top-down traversal tag allocation strategy is much more efficient (around five times quicker) than the existing tag allocation strategy, while maintaining a similar allocation accuracy.

D. STRUCTURE OF THE PAPER

The rest of this paper is organized as follows. The following Section II reviews related work. In Section III, we give some preliminaries, and we recall the state-of-the-art similarity measures for content-based personalization. Section IV then describes the new ontological similarity measure, while Section V deals with personalized recommendation based on this ontological similarity. In Sections VI and VII we provide our experimental results and some concluding remarks, respectively.

II. RELATED WORK

In this section we review related work on personalization, folksonomy-based personalization, personalization using ontologies, and word sense disambiguation.

A. PERSONALIZATION

Achieving personalized Web search and recommendation that adequately consider the searcher's personal attributes and preferences is a very important topic of research [30]. As the first step of personalization, users' preferences and interests are required in order to build a user profile, which is then used for personalization by re-ranking [5] (i.e., results are re-ranked according to the searcher's profile such that personally relevant results appear higher in the search result list) or query expansion [31] in search (i.e., a user's query is expanded, based on the user profile to reflect his/her particular interests).

In general, a user's preferences or interests can be obtained by leveraging information that is either explicitly or implicitly provided. *Explicit* user preferences are usually offered by the searcher directly by filling in relevant information or providing positive or negative relevance feedback for the given results [6]. As explicit user preferences require extra efforts from users, it may result in a lower quality experience, since the amount of information that is available directly depends on the user's willingness to provide it directly. Hence, as an effortless alternative for users, many approaches have already been proposed to learn user preferences from their *implicit* activities on the Web, such as the query history [7]–[10], the browsing history [11], the users' current tasks [12] or intent [13], and even eye-tracking during the search session [14], which are then used for the personalization of search results.

B. FOLKSONOMY-BASED PERSONALIZATION

One drawback of learning user preferences by aggregating and mining users' online activities is that these methods will inevitably result in problems related to privacy [32]: due to the great variety of online activities that users engage in every day, Web logs usually contain sensitive information such as home addresses, medical records, bank account information, or even social security numbers. Therefore, as a privacy-enhanced personalization technique, *social annotations* (also known as *folksonomies*) have recently been at the forefront of research efforts to enhancing Web search. This is because social annotations are the results of users' public activities and contain little sensitive information about their creators.

Existing research on using social data in information retrieval includes folksonomies [17], [33], [34] and more general social media data across services [35]. Early approaches for using social data to enhance information retrieval are the Adapted PageRank methods, in particular SocialSimRank, SocialPageRank [36], and Topic-Driven SocialRank [37], which are used to compute similarities between users and their social networks to enhance Web search. A similar method, called ScenticRank, which makes use of sentiment information in addition to social graphs, has also been proposed [38]. Furthermore, [39] describes a new approach for acquiring precise resource descriptions based on social annotations available in the social bookmarking service, while [40] uses WordNet to determine the meaning of target tags and their synonyms. Web search results are enhanced using personalized social document representations in [41].

There are also several works that survey related literature. In particular, in [42], Campana and Delmastro survey recommender systems designed and implemented for online and mobile social networks. In [43], it is given an overview of content-based recommender systems, collaborative filtering systems, hybrid approaches, and memory- and model-based algorithms and features of collaborative tagging that are generally attributed to their success and popularity; a model for tagging activities and tag-based recommender systems is also presented. Context-aware recommender systems and their relationship with social search are reviewed in [44] and [45].

As for folksonomy-based personalized search and recommendation, the vector space model (VSM) [25] is the most widely adopted data model for content-based personalization solutions [15], [18], [20], [22]. VSM is a general model used in information retrieval where the profile of a user (resp., a document) is mapped to a weighted vector in a universal term space. Consequently, in order to achieve personalized search and recommendation, VSM-based methods [15], [18], [20], [22] usually first use folksonomies to model the document and user profiles as weighted vectors whose dimensions are tags and whose values in each dimension are the corresponding tag weights. Then, online documents are re-ranked according to a personalized ranking based on the similarity between the two profiles. Specifically, [15] and [18] propose to use cosine similarity of folksonomy-based user and document profiles to personalize search and recommendation results on the Social Web, respectively. The work in [15] is then extended by [22], where a social matching score is introduced to better summarize the content of a document and to add further information for social resources with very little textual content (e.g., videos and images). Later, [20] proposed to use scalar similarity as the metric for personalization, which eliminates the user and the document profile length normalization factors in the cosine similarity to avoid penalizing popular documents.

The similarity metrics used in all these works [15], [18], [20], [22] are either cosine or scalar similarity, which can only match tags literally and thus leverage the semantics of tags only to a limited extent. Due to uncontrolled vocabularies, social tags are usually ambiguous, which leads to inaccurate similarity values and greatly degrades the performance of content-based tag-aware recommendation systems. A solution to this problem is to apply clustering in the tag space [19], such that redundant tags are aggregated; this also reduces ambiguities, since tags in the same cluster share the same meaning. But tag clustering is usually time-consuming in practice, so [21] further proposes a solution to use autoencoders to solve this problem, due to their capability to extract abstract representations [46]. Finally, [28] proposes a deep-semantic similarity-based personalized recommendation (DSPR) solution, which maps the tag-based user and item profiles to an abstract deep feature space, where the deep-semantic similarities between users and their target items (resp., irrelevant items) are maximized (resp., minimized). As they are the state-of-the-art machine learning solutions for the tag ambiguity problem, these three methods are used as baselines in our experimental evaluation.

Besides content-based solutions, folksonomy-based recommendation can also be achieved by using collaborative filtering and graph-based ranking approaches. Specifically, [47] proposes to use model-based collaborative filtering based on probabilistic matrix factorization for recommendation using folksonomies. In addition, a graph-based ranking solution, called *FolkRank* is proposed in [33], which extends the PageRank algorithm to folksonomies where tags, users, and documents are treated as nodes and are connected via assignments, and then a weight passing scheme is used to derive the importance of these nodes. This work focuses on content-based solutions—collaborative filtering and graph-based solutions will be explored in future work.

C. ONTOLOGIES AND CONCEPT SIMILARITY

An ontology is a formal specification of the types, properties, and interrelationships of the entities for a particular domain of discourse [29]. Since ontologies are usually constructed based on the consensus of domain experts, they are highly reliable structured knowledge bases, having a wide range of applications. However, the number of domain ontologies are huge, and their content is various and usually inconsistent [48]; thus, in order to integrate ontologies from different sources, many metrics have been proposed to quantify the semantic relevance between concepts in the ontologies.

In [49], Rada *et al.* propose the use of the length of the shortest path between two concepts to measure their likeness, while the solutions proposed by [50] and [51] use the relative depths of two concepts and their least common ancestor. Similarly, Jiang and Conrath [52] and Lin [53] propose to use the information content of the two compared concepts and their least common ancestor to calculate the similarity between the two concepts. In [54], Li *et al.* proposes to combine the shortest path with the depth of ontology information to estimate the similarity non-linearly. As for ontologies containing non-taxonomic semantic links, Hirst and St-Onge [55] extend the taxonomic solution and take into account the number of times that the link direction changes such that the more changes in relation direction on the shortest path, the lower the likeness. On the other hand, here we do not propose new concept similarity metrics, but rather to exploit ontology-based concept similarities to solve a novel practical problem (inaccurate profile matching) in personalization.

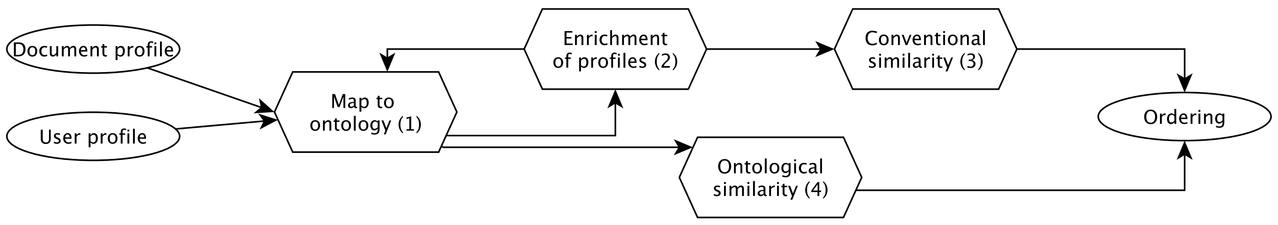
Document profile
User profile
Map to ontology (1)
Enrichment of profiles (2)
Conventional similarity (3)
Ontological similarity (4)
Ordering

FIGURE 2. Difference between our approach and the state of art.

D. CONNECTING FOLKSONOMIES AND ONTOLOGIES

There is some work that tries to connect folksonomies with ontologies and proposes strategies to solve the co-occurrence problem in tag-to-ontology mappings. Specifically, [56] takes the Open Directory Project (ODP) taxonomy[1] as the underlying ontology (since each concept in ODP contains a set of references to related online documents) to solve the multiple occurrence problem, they propose to view these references as context, and define the most appropriate concept as the one associated with the highest number of references. However, this method has some limitations that make its use restricted: first, tag allocation is static, i.e., the most appropriate concept is fixed for each tag; second, not all ontologies contain this kind of statistical information as in the case of ODP. A more dynamic and adaptive strategy is proposed by Angeletou *et al.* in [57] based on Wu and Palmer's [50]similarity from: it first maps the given tag to all possible candidate concepts and then uses the contextual information of its tag set to identify the most relevant concept by computing the similarity of all combinations of tags in this tag set using Wu and Palmer's similarity in the ontology. Given two tags with more than one matching concept for each of them, this method selects the two concepts having the highest similarity value above a predefined threshold. In our work, as an alternative method to adaptively map tags to ontologies, the strategy in [57] is considered as the competitor of our proposed top-down strategy in the evaluation.

There are also some studies that integrate ontologies with folksonomies for personalized search or recommendation. For example, the work in [56] investigates how to use domain ontologies to model semantically more intuitive user profiles, and Movahedian and Khayyambashi [58] propose a folksonomy-based recommender system based on user and document profiles that are both semantically enriched by an external knowledge base. Taxonomies are also leveraged in [59] to generate semantic resource profiles for personalized social tag recommendation. In summary, the purpose of the above works is to solve the *profile modeling problem* in personalized search by using ontological information to model semantically enriched user or resource profiles—*however, the metrics used in the above works to compute the similarity between the enriched profiles are still the conventional ones*. Therefore, the personalized ordering is done by following the procedure (1) → (2) → (3), as shown in Figure 2. On the other hand, ontologies in our work are used to solve the *tag ambiguity problem* in personalization, where an ontological similarity measure is proposed to leverage domain ontologies to disambiguate tags and compute semantically more accurate similarity scores between user and document profiles (so, our personalized ordering procedure is (1) → (4) in Figure 2). Thus, we actually leverage ontological information to solve different problems in the process of personalization, and the techniques in our work can be orthogonally combined with all the above works to obtain a further enhanced personalization (resulting in a hybrid ordering procedure (1) → (2) → (1) → (4)).

E. TAG DISAMBIGUATION AND WORD SENSE DISAMBIGUATION TO WIKIPEDIA

There are some works that aim to disambiguate tags in folksonomies. Cattuto *et al.* [26] propose to measure the relatedness between tags using a tag-tag co-occurrence graph, called FolkRank, and three distributional measures with three different vector space representations (tag context, document context, and user context) of tags, where cosine similarity is used to estimate the similarity between vectors. This allows to disambiguate synonyms and homonyms using their most related tags. Furthermore, the distributional measures are extended with different aggregation methods (e.g., projection and macro-aggregation) in [27], where other similarity metrics (e.g., Jaccard, Dice, and mutual information) are also introduced. To evaluate the performance of our ontology-based solution in disambiguating tags, these existing approaches will be used as baselines in our experiments.

Using ontologies for tag disambiguation is also similar to the problem of *word sense disambiguation to Wikipedia*, or simply *disambiguation to Wikipedia (D2W)* [60]. Generally, the goal of D2W is to disambiguate a set of explicitly identified substrings, e.g., words or phrases, in a given document by mapping each substring to a Wikipedia article. D2W also suffers from the multiple occurrence problem: a substring may be mapped to multiple Wikipedia articles. To solve this problem, many works [60]–[63] have been proposed to use the unambiguous substrings (substrings mapped to a unique Wikipedia article) in the same document and their mapped Wikipedia articles as semantic context to disambiguate the ambiguous substrings. In order to improve accuracy, Li *et al.* [62] also introduce a

[1]http://www.dmoz.org/

confidence score in disambiguation and use high-confidence disambiguated substrings as additional context to improve the disambiguation accuracy of low-confidence disambiguated substrings.

Therefore, it seems to be possible to apply D2W solutions for tag disambiguation by using Wikipedia as the underlying ontology (considering articles as concepts, connected by hyperlinks or categories) and then use the uniquely matched co-occurring tags as context to disambiguate tags. However, in folksonomies, since each user usually annotates a document with only a few (1–3) tags, and the portion of tags that can be uniquely matched to the ontology is usually quite small (e.g., in our experiments, only 10.2% of tags in the Delicious dataset are uniquely matched to ODP, and more than half of the user and document profiles in our experiments have less than 3 uniquely matched tags), using only uniquely matched tags usually cannot provide sufficient context for tag disambiguation. Therefore, in this work we propose to use all tags that can be matched in the ontology as context to ensure sufficient contextual information for disambiguation.

III. PRELIMINARIES

We now briefly recall folksonomies and the vector space model (VSM). We then formalize content-based personalized recommendation in the context of these definitions; finally, we recall two VSM-based similarity measures for personalization.

A. FOLKSONOMIES AND VECTOR SPACE MODEL

A *folksonomy* is a tuple $\mathcal{F} = (U, T, D, A)$ [33], where $T = \{t_1, \ldots, t_m\}$ is the set of *tags* that comprise the vocabulary expressed by the folksonomy; $U = \{u_1, \ldots, u_k\}$ and $D = \{d_1, \ldots, d_k\}$ are the sets of *users* and *documents* that annotate and are annotated with the tags of T, respectively, and $A \subseteq U \times T \times D$ is the set of assignments (u, t, d) of each tag t to a document d by a user u. For instance, in the example in Figure 1, we have $U = \{u_1, u_2, u_3\}$, $T = \{Apache, Jaguar, Mammal\}$, $D = \{d_1, d_2\}$, and $A = \{(u_1, Apache, d_1), (u_1, Jaguar, d_1), (u_2, Jaguar, d_2), (u_2, Mammal, d_2), (u_3, Mammal, d_2)\}$.

The vector space model (VSM) [25] is a general model used in information retrieval where the profile of a user (resp., document) is mapped to a weighted vector $\vec{p}_u$ (resp., $\vec{p}_d$) in a term space. As for the case of folksonomies, the terms are tags, while the weights of terms are based on the frequency of tags. So, the profile of a user u is defined as $\vec{p}_u = \{(t_1, w_1), \ldots, (t_i, w_i), \ldots, (t_{m_u}, w_{m_u})\}$, where every t_i is a tag, every w_i is the number of times that u annotates documents using t_i, and m_u is the number of different tags used by u. Similarly, the profile of a document d is defined as $\vec{p}_d = \{(t_1, w_1), \ldots, (t_j, w_j), \ldots, (t_{m_d}, w_{m_d})\}$. In the example in Figure 1, we have that $\vec{p}_{u_1} = \{(Apache, 1), (Jaguar, 1)\}$, $\vec{p}_{u_2} = \{(Jaguar, 1), (Mammal, 1)\}$, $\vec{p}_{u_3} = \{(Mammal, 1)\}$, $\vec{p}_{d_1} = \{(Apache, 1), (Jaguar, 1)\}$, and $\vec{p}_{d_2} = \{(Jaguar, 1), (Mammal, 2)\}$.

B. CONTENT-BASED PERSONALIZED RECOMMENDATION IN FOLKSONOMIES

Using the above concepts, *content-based personalized recommendation in folksonomies* can then be formulated as follows. Given a user u, the system produces a ranked recommendation list $\tau = [d_1 \geqslant d_2 \geqslant \cdots \geqslant d_k]$ of all documents in D such that $d_i \geqslant d_j$ if and only if $Rank(d_i, u) \geqslant Rank(d_j, u)$. Here, $Rank(d, u)$ is a ranking function measuring how relevant document d is to user u:

$$Rank(d, u) = Sim(\vec{p}_u, \vec{p}_d). \tag{1}$$

Clearly, the personalization performance depends greatly on the effectiveness of the adopted similarity measure in $Sim(\vec{p}_u, \vec{p}_d)$.

C. SIMILARITY MEASURES FOR PERSONALIZATION

We now briefly discuss two state-of-the-art VSM-based similarity measures for personalization in Web search, namely *cosine* and *scalar similarity*—these are used later as baseline methods in our experimental evaluation.

The most widely adopted similarity measure in personalization is cosine similarity [15], [18], [22]. Following the VSM approach, the *cosine similarity* between a user profile $\vec{p}_u$ and a document profile $\vec{p}_d$, denoted $Sim_{Cosine}(\vec{p}_u, \vec{p}_d)$, is defined as follows:

$$Sim_{Cosine}(\vec{p}_u, \vec{p}_d) = \frac{\sum_{i=1}^{n}(w_{u_i} \cdot w_{d_i})}{\sqrt{\sum_{i=1}^{n}(w_{u_i})^2} \cdot \sqrt{\sum_{i=1}^{n}(w_{d_i})^2}}, \tag{2}$$

where $n = |T|$ is the number of different tags, and every w_{u_i} (resp., w_{d_i}) is the weight of tag t_i in profile $\vec{p}_u$ (resp., $\vec{p}_d$).

Later, Vallet *et al.* [20] proposed the so-called scalar similarity, which is similar to the cosine similarity, except that it eliminates the user and the document profile length normalization factors. Formally, the *scalar similarity* between a user profile $\vec{p}_u$ and a document profile $\vec{p}_d$, denoted $Sim_{Scalar}(\vec{p}_u, \vec{p}_d)$, is defined as follows:

$$Sim_{Scalar}(\vec{p}_u, \vec{p}_d) = \textstyle\sum_{i=1}^{n}(w_{u_i} \cdot w_{d_i}), \tag{3}$$

where n and all w_{u_i} and w_{d_i} are defined as above.

IV. ONTOLOGICAL SIMILARITY

As discussed in Section I, the two state-of-the-art similarity measures apply literal matching of tags; due to ambiguous tags in folksonomies, using these two measures may result in inaccurate similarity values between the profiles. To overcome this problem, we propose an *ontological similarity* measure that uses ontologies to disambiguate tags and leverages concept similarity in ontologies to quantify the semantic relevance of tags in profiles.

The computation of this ontological similarity measure mainly consists of three steps: (i) *tag allocation:* the tag in a profile is first semantically matched to a single concept in an ontology; (ii) *computing concept similarities:* we then use the semantic relevance of concepts in ontologies to estimate the semantic similarity of corresponding tags; and (iii) *computing* *ontological similarities:* the ontological similarity of profiles is obtained by integrating the concept similarities with the conventional cosine and scalar similarities.

In this work, we generally refer to the Open Directory Project (ODP) taxonomy[2] as an example of an underlying domain ontology. ODP is one of the largest and most comprehensive human-edited directories of the Web, and is widely adopted by many other research works in Web personalization [56]. However, the methods proposed here are not restricted to ODP and can also be used with other ontologies, such as WordNet.

A. TAG ALLOCATION

Ontologies can be seen as a directed graph in which concepts are interrelated mainly via subsumption (is-a) relationships. Therefore, concepts in the ODP ontology are organized hierarchically as a *taxonomy*, forming a tree structure—it can thus be seen as an open directory of the Web.

During tag allocation, due to homonyms, it may be possible to map a certain tag to multiple concepts (called *multiple occurrence*); therefore, solutions are needed to find the most relevant concept as the matching concept of this tag. One existing method [57] is to first map the given tag to all possible candidate concepts and then use the contextual information of its profile to identify the most relevant concept by computing the similarity of all combinations of tags in this profile using the Wu and Palmer similarity in the ontology. Given two tags with more than one matching concept for each of them, this method selects the two concepts having the highest Wu and Palmer similarity value. Given an ontology with r concepts and a profile with size m, assuming the computational cost of the Wu and Palmer similarity to be $O(1)$, the complexity of tag allocation for each profile is up to $O(m^2 \cdot r^2)$ in the worst case, because there are $O(m^2)$ combinations of tag pairs, and each pair costs $O(r^2)$ in the worst case.

Here, we thus develop a more efficient two-step top-down disambiguation, which runs in time $O(m \cdot r^2)$ ($O(m \cdot r \cdot \log(r))$, if the ontology is a balanced tree) in the worst case. Similarly to [57], the first step of this top-down strategy is to map the tags to all possible concepts in the ODP taxonomy; and then, for each tag with more than one matching candidate concept, we apply a top-down traversal from the root of the tree structure, to iteratively narrow down the number of candidate concepts by using the statistical information of the matching concepts of other tags in the same profile as context. The algorithm for tag allocation is provided in Algorithm 1. Intuitively, it disambiguates tags in a given profile by mapping the tags to concepts in the domain ontology and using the matching concepts of other tags as context to disambiguate the tags with ambiguity. We now describe the detailed process.

Mapping Step (Lines 4–11): Given a profile, for each tag, we traverse the whole taxonomy to identify all possible candidate concepts in the hierarchy (Lines 6–8), where the matching is based on string equivalence; if there is no candidate concept found for a tag, we call this tag *unmatched tag*; if only one candidate concept is found for a tag, it is directly selected as the *matching concept* for this tag (Lines 10–11); if more than one concept is found (multiple occurrence problem), we mark these concepts as *candidate matching concepts* for this tag and eliminate its ambiguity in the second step. To help disambiguation, we increase the concept weights

[2]http://www.dmoz.org/

ALGORITHM 1: TagAllocation($\vec{p}$, Ont)

```
Input: profile (either user or document profile): $\vec{p}$, domain
       ontology: $Ont$
Output: $hashmap_{match}$ — a hashmap saving the matching
        concepts of tags in $\vec{p}$
1  foreach $concept\ c \in Ont$ do
2      $cw_c = 0$; // Initialize the concept weight of $c$, denoted $cw_c$
3  $hashmap_{match} = \emptyset$;
4  foreach $(tag\ t, weight\ w_t) \in \vec{p}$ do
5                                              // Mapping step
6      $list_t = \emptyset$;
7      foreach $concept\ c \in Ont\ that\ matches\ t$ do
8          $list_t.add(c)$;
9          $UpdateConceptWeights(c, w_t, Ont)$;    // Algorithm 2
10     if $SizeOf(list_t) = 1$ then
11         $hashmap_{match}.add(t, c)$;    // set $c$ to be the matching
           concept of $t$;
12 foreach $(t, w_t) \in \vec{p}$ with $SizeOf(list_t) > 1$ do
13                                       // Disambiguation step
14     $c_s = Root(Ont)$;
15     do
16         $MaxWeight = 0$;
17         $MaxConcept = $ null;
18         foreach $concept\ c' \in ChildrenOf(c_s)$ do
19             if $DescendantsOf(c') \bigcap list_t \neq \emptyset$ then
20                 if $cw_{c'} \leqslant MaxWeight$ then
21                     $list_t.remove(\{DescendantsOf($
22                     $c') \bigcap list_t\})$
23                 else
24                     if $MaxConcept \neq$ null then
25                         $list_t.remove(\{DescendantsOf($
26                         $MaxConcept) \bigcap list_t\})$;
27                     $MaxWeight = cw_{c'}$;
28                     $MaxConcept = c'$;
29         $c_s = MaxConcept$        // update $c_s$ to the max child
           concept
30     while $SizeOf(list_t) > 1$;
31     $hashmap_{match}.add(t, list_t[0])$;  // set the only concept left
       in $list_t$ to be the matching concept of $t$
32 return $hashmap_{match}$.
```

ALGORITHM 2: UpdateConceptWeights(c, w, Ont)

```
1  $cw_c += w$;   // increase the concept weight of concept $c$ by $w$
2  foreach $concept\ c_i \in AncestorsOf(c)$ do
3      $cw_{c_i} += w$;   // increase the concept weight of concept $c_i$
       by $w$
```

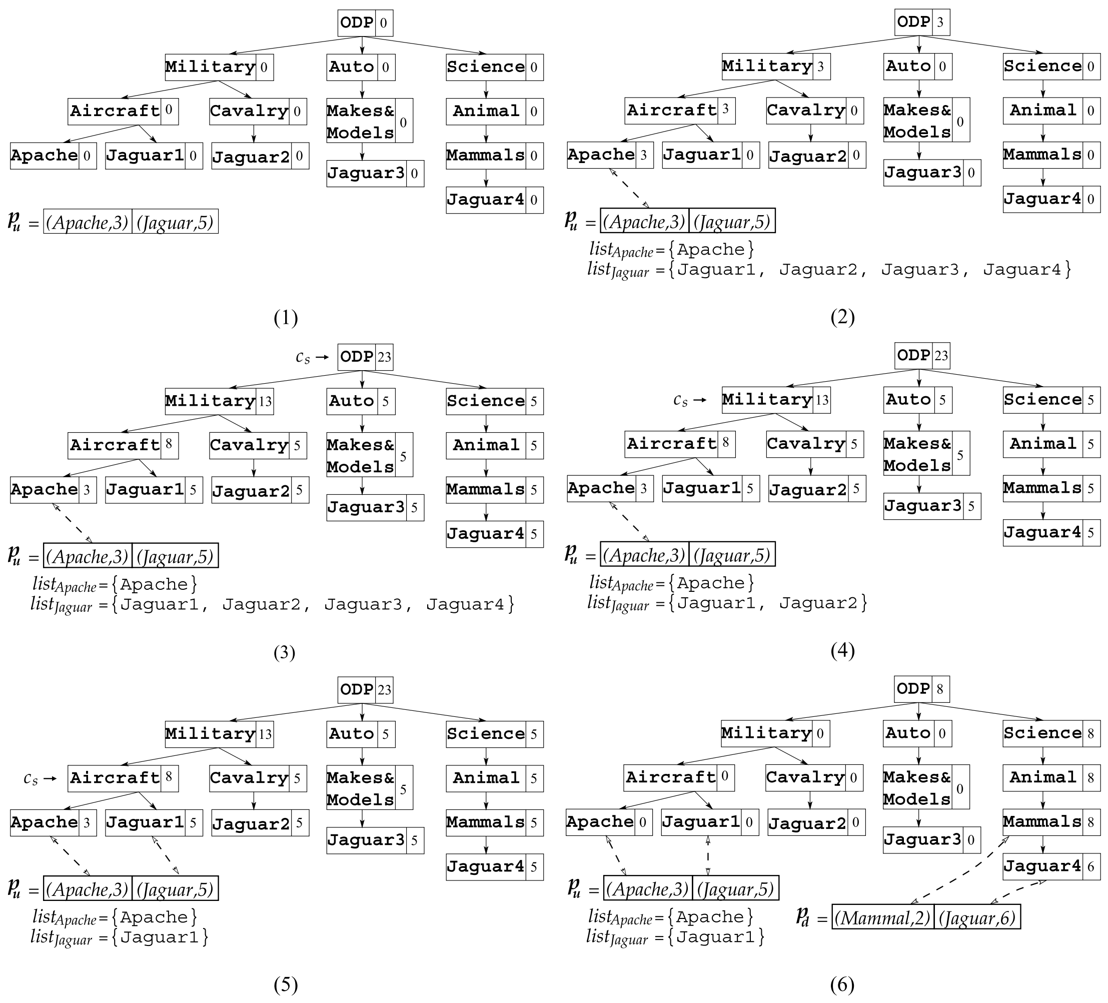

FIGURE 3. Allocation of tags in the ODP taxonomy.

(denoted *cw*) of the (candidate) matching concept and all its ancestors by the corresponding tag weights (denoted *w*).

Disambiguation Step (Lines 12–29): After the mapping of all tags, a top-down disambiguation process is invoked to resolve the multiple occurrence problem; specifically, for each tag with multiple candidate concepts, we first (i) assign the root of the taxonomy to a concept variable c_s (Line 14), and (ii) compare the concept weights of the child concepts of c_s that contain at least one candidate concept as its descendant (Lines 18–26), then (iii) select a child concept with highest concept weight as new c_s (Line 27) and unmark the candidate concepts that are not the descendants of the new c_s (Lines 21–22 and 25–26); we repeat (ii) and (iii) until only one candidate concept is left and select it as the matching concept of this tag (Lines 28–29). Note that if multiple child concepts have a same highest weight, the one first seen by the iterator is selected.

The following example illustrates this tag allocation algorithm and its advantage in dynamically allocating concepts to tags according to their semantics.

Example 1: Consider a folksonomy extending the one of Figure 1, and let *u* be a user who is a military enthusiast (inferred from the user profile) that has a user profile $\vec{p}_u$ containing two tags, *Apache* and *Jaguar*, with weights 5 and 3, respectively—i.e., $\vec{p}_u = \{(Apache, 3), (Jaguar, 5)\}$.

We would like to allocate these two tags to concepts in the ODP ontology as shown in Figure 3. According to Algorithm 1, we first initialize all concept weights in the ontology to 0 (as shown in Figure 3 (1)). Then, we traverse the whole hierarchy to find the corresponding candidate concepts of both tags and save them in the respective lists; since there is only one candidate concept, *Apache*, in the list $list_{Apache}$, we directly select this concept as the (direct) matching concept of the tag *"Apache"* and also increase the concept weights of *Apache* and its ancestors by 3 (as shown in Figure 3 (2)).

However, as for the tag *"Jaguar"*, there are four candidate concepts found in the list $list_{Jaguar}$ (numbered for reference purposes). So, the size of $list_{Jaguar}$ is larger than 1. Consequently, we also increase the concept weights of all *Jaguar* concepts and their ancestors by 5 and invoke the disambiguation procedure to solve the multiple occurrence problem in the second step. The disambiguation uses concept weights as context to incrementally narrow the size of $list_{Jaguar}$. This process is started by setting the root concept of the *ODP* ontology as the current concept variable c_s (as shown in Figure 3 (3)). Then, we compare the concept weights of c_s's child concepts, of whom at least one of these four candidate concepts are descendants. As the child concept *Military* has the highest concept weight (13 vs. 5), we remove *Jaguar3* and *Jaguar4*, which are not the descendants of *Military*, from the candidate list, $list_{Jaguar}$, and reset concept *Military* as the new c_s (as shown in Figure 3 (4)). As there are still two distinct candidate concepts (*Jaguar1* and *Jaguar2*) in $list_{Jaguar}$, the do-while loop in Algorithm 1 continues to compare the concept weights of *Military*'s child concepts; at this iteration, *Aircraft* defeats *Cavalry*; so, we further remove *Jaguar2*, and now $list_{Jaguar}$ contains only one concept. We thus select the only remaining concept *Jaguar1* as the matching concept of the tag *"Jaguar"* (as shown in Figure 3 (5)).

Similarly, given an online document d about animals, with profile $\vec{p}_d = \{(Mammal, 2), (Jaguar, 6)\}$, the concept *Mammals* can be directly identified as the matching concept of tag *"Mammals"*. However, after the disambiguation process, *Jaguar4* is selected as the matching concept of the same tag *"Jaguar"* in profile $\vec{p_d}$ (as shown in Figure 3 (6)). This example shows the advantage of our tag allocation methodology in dynamically allocating tags to the semantically most suitable concepts. ■

The following result states the computational cost of tag allocation with Algorithm 1 in the worst case, in general and when the underlying taxonomy is a balanced tree. Note that the case where the tree structure of the taxonomy is very unbalanced (which is the reason for the higher worst-case complexity in general) is very rare in practice.

Proposition 1: Given a taxonomy with r concepts and a (user or document) profile of size m, Algorithm 1 runs in time $O(m \cdot r^2)$ in the worst case. If the taxonomy is a balanced tree, then Algorithm 1 runs in time $O(m \cdot r \cdot \log r)$ in the worst case.

Proof: The for loop in Line 1 takes time $O(r)$ (for the initialization). The next loop in Line 4 (for the mapping) consists of $O(m \cdot r)$ steps for general taxonomies, and $O(m \cdot \log r)$ steps if the underlying taxonomy is a balanced tree. More precisely, two for loops in Lines 4 and 7 result in $O(m)$ iterations, if concepts are accessible in $O(1)$, e.g., saved in a hashmap, and the number of matching concepts is treated as a constant, while the worst-case cost of the concept weight update in Line 9 is $O(r)$ for general taxonomies and $O(\log r)$ for balanced-tree taxonomies.

Both are dominated by the disambiguation step, whose time complexity is $O(m \cdot r^2)$ (resp., $m \cdot O(r \log r)$ for balanced-tree taxonomies) in the worst case: Line 12 is done in $O(m)$ steps (outer for loop in Line 12), each of them needs $O(r^2)$ in the worst case for general taxonomies and $O(r \log r)$ for balanced-tree taxonomies. To see that this is the case, note that the do loop in Line 15 together with the nested for loop in Line 18 results in $O(r)$ (resp., $O(\log r)$ for balanced-tree taxonomies) iterations in the worst case. This is because the do loop iteratively selects a concept in each level of the taxonomy (from the root to a leaf in the worst case) as c_s, and the nested for loop iteratively traverses all children of the current c_s; consequently, these two loops together result in a top-down traversal of the taxonomy following only one branch of the tree, and each concept in this branch will be traversed at most once, such that the number of iterations for the traversal is $O(r)$ (resp., $O(\log r)$ for balanced-tree taxonomies) in the worst case. However, in each iteration of the for loop, we also have to obtain the descendant list of the current traversed concept c' (i.e., $DescendantsOf(c')$ in Line 19), whose computational cost is $O(r)$ in the worst case. So, each iteration of the outer for loop in line 12 takes $O(r^2)$ (resp., $O(r \cdot \log r)$ for the balanced-tree case) in the worst case, and the total worst case time complexity for Algorithm 1 is thus $O(m \cdot r^2)$ (resp., $O(m \cdot r \cdot \log r)$). □

B. COMPUTATION OF ONTOLOGICAL SIMILARITY

After the tag allocation, we are able to compute the ontological similarity between a given user profile and a given document profile. For a matching concept c_u of a user profile tag t_u, we first define its *nearest concept* c_d as the matching concept of a tag t_d in $\vec{p}_d$ that satisfies the following two conditions: (i) the least common ancestor (*lca*) of c_u and its nearest concept c_d must be a descendant of (or the same as) the *lca* of c_u and any other matching concept of a tag in $\vec{p}_d$; and (ii) if there exist other matching concepts of a tag in $\vec{p}_d$ whose *lca* with c_u is the same as that of c_d, these concepts must be in a lower level of the hierarchy than (or the same level as) c_d. The intuition behind this definition is that concept specifications in a taxonomy are recursively refined, so the defined nearest concept c_d is the semantically closest one to c_u compared to any other matching concept of a tag in $\vec{p}_d$.

Then, the ontological similarity can be obtained by the following steps: (1) For each matching concept c_u of a user profile tag t_u, we use its semantic relevance to the nearest concept c_d of a tag t_d to estimate the semantic similarity of t_u and t_d. We call this similarity measure *concept similarity* and use breadth-first search to find the nearest concept in the taxonomy bottom-up. (2) We then integrate the resulting concept similarities with the cosine or scalar similarities to get the final ontological similarity.

We summarize the process of computing the cosine-based ontological similarity in Algorithm 3. Generally, given a user profile p_u and a document profile p_d, this algorithm estimates the semantic similarity between the two given profiles by using the concept similarity between the matching concepts of the tags in p_u and their nearest concepts in p_d. Intuitively, this ontology-based similarity will be able to achieve semantically more accurate similarity values, because homonyms (resp., synonyms) are mapped to semantically very different (resp., close) concepts in the ontology, based on the different (resp., similar) profile context, resulting in a low

ALGORITHM 3: OntologicalSimilarity$(\vec{p}_u, \vec{p}_d, hashmap_{p_u}, hashmap_{p_d}, Ont)$

```
   Input: User profile: $\vec{p}_u$; document profile: $\vec{p}_d$; domain
          ontology: $Ont$; hashmap saving matching concepts in
          $p_u$ and $p_d$: $hashmap_{p_u}$ and $hashmap_{p_d}$
   Output: $ontSim_{Cosine}$ — cosine-based ontological similarity
           of $\vec{p}_u$ and $\vec{p}_d$
1  foreach concept $c \in Ont$ do
2      $counter_c = 0$;  // counts the number of times $c$ is selected
       as the nearest concept
3  $sum_u = 0$; $sum_d = 0$; $Similarity = 0$;
4  foreach $(t_u, w_u) \in \vec{p}_u$ do
5      $sum_u += (w_u)^2$; $c_u = hashmap_{p_u}.get(t_u)$;     // $w_u$ is the
       weight of tag $t_u$ in $\vec{p_u}$, and $c_u \in Ont$ is the matching
       concept of $t_u$
6      if $c_u \neq null$ and $SizeOf(hashmap_{p_d}) > 0$ then
7          $lca = c_u$; $lca' = \texttt{null}$; $c_d = \texttt{null}$;   // $lca$ is found
           bottom-up from $c_u$ to root
8          do
9              foreach concept $c' \in Subtree(lca)$ with $c' \notin$
               $Subtree(lca')$, in breadth-first exploration do
10                 if $c'$ is the matching concept of a tag $t_d$ with
                   $(t_d, w_d) \in \vec{p_d}$ then
11                         // $w_d$ is the weight of tag $t_d$ in $\vec{p_d}$
12                     $c_d = c'$;
13                     $counter_{c_d}++$;
14                     break;
15             if $c_d = \texttt{null}$ then
16                 $lca' = lca$;
17                 $lca = Parent(lca)$;
18         while $c_d = \texttt{null}$;
19         $conSim = ConSim(c_u, c_d)$ according to Equations 4
           or 5
20     else
21             // when $t_u$ is unmatched tag, or no tag in $\vec{p_d}$ has
           matching concept
22         if $t_u = t_d$ for a $(t_d, w_d) \in \vec{p}_d$ then
23             $conSim = 1$
24         else
25             $conSim = 0$;
26     $Similarity += w_u \cdot w_d \cdot conSim$;
27 foreach $(t_d, w_d) \in \vec{p}_d$ do
28     $k = 1$;
29     if $counter_{c_d} > 1$ then
30         $k = counter_{c_d}$;
31     $sum_d += k \cdot (w_d)^2$;
32 $ontSim_{Cosine} = \frac{Similarity}{\sqrt{sum_u} \cdot \sqrt{sum_d}}$;      // according to Equation 6
33 return $ontSim_{Cosine}$.
```

(resp., high) ontological similarity between homonyms (resp., synonyms) in different profiles.

Each step of Algorithm 3 is explained in detail as follows:

(1) *Initialization* (Lines 1–2): We first set $counter_c = 0$ for each concept c in Ont (Lines 1–2). The variable $counter_c$ is used to record the number of times that c is selected as the nearest concept of a tag in the user profile.

(2) *Concept Similarity Computation* (Lines 4–26): For each tag t_u with a weight w_u in $\vec{p}_u$, we check whether it has a matching concept c_u in Ont, and whether at least one tag in $\vec{p}_d$ has a matching concept (Line 6). If not, we set the concept similarity to 1, if t_u is also in $\vec{p}_d$, and to 0, otherwise (Lines 20–25). If a matching concept is found, we conduct a bottom-up search to find its nearest concept c_d and lowest common ancestor concept (lca), and then compute the concept similarity of c_u and c_d, using the measures presented in Section IV-B.1 (Lines 7–19). The detailed search process is as follows:

   (i) We first assume lca is c_u and use lca' to store the lowest common ancestor selected in the last iteration (Line 7).

   (ii) For each concept c' in the subtree with lca as root and also not in the subtree with lca' as root, we check whether c' is the matching concept of a tag t' in $\vec{p}_d$. If so, we select c' as the nearest concept c_d of c_u, increase $counter_{c_d}$ by one, and stop the search (Lines 9–14). To make sure that the first found matching concept of $\vec{p}_d$ is the nearest concept of c_u as defined above, we use breadth-first search.

   (iii) After the traversal of the subtree, if there is no valid c_d found, we set the value of lca' to be lca and use the parent concept of the current lca as the new lca (Lines 15–17). Steps (ii) and (iii) are repeated until a valid c_d is found (Line 18).

(3) *Ontological Similarity Computation* (Lines 27–32): Finally, the ontological similarity is computed according to the proposed metric in Equation 6 in Section IV-B.2, which integrates the concept similarity with the traditional cosine similarity (line 32).

The algorithm of computing the scalar-based ontological similarity is very similar to Algorithm 3, except that it does not require the normalization factors in Lines 3, 5, and 27–31, but returns directly the weighted sum of concept similarities in Line 26 as the scalar-based ontological similarity (cf. Equation 7).

The following result states the cost of computing ontological similarities with Algorithm 3 (when the matching between tags and concepts is also a part of the input in the form of an assignment of concepts to tags and vice versa).

Proposition 2: Given a taxonomy with r concepts, user profile of size m_u, and a document profile of size m_d, Algorithm 3 runs in time $O(m_u \cdot (r + m_d))$ in the worst case.

Proof: The for loop in Line 1 takes $O(r)$ steps. The loop in Line 4 consists of $O(m_u)$ steps, each of which in the worst case consists of $O(r)$ steps. This is because the statement in Line 9 makes sure that each concept in the ontology will be traversed at most once in the do-while loop in Lines 8–18. In addition, checking whether t_u is also contained in $\vec{p}_d$ in Line 22 needs $O(m_d)$. Finally, the loop in Line 27 takes time $O(m_d)$. The total worst-case running time is therefore $O(m_u \cdot (r + m_d))$. □

1) CONCEPT SIMILARITY

To measure the similarity between two concepts, we make use of two effective metrics based on two important dimensions of taxonomies, i.e., the concepts' relative depth and their shortest path, whose effectiveness has been well-proven in existing works [51], [54], [64]. However, the use of our proposed technique is not restricted to these two path-based metrics; other metrics, such as the information-content-based Jiang-Conrath similarity [52] and the Lin similarity [53] can also be adopted.

The relative depth in the taxonomy of the concepts is an important dimension, because concept specifications are recursively refined, so: (i) concepts in upper levels are less similar than those in lower levels; and (ii) concepts subsumed by an upper common ancestor are more different than those subsumed by a lower ancestor. Consequently, a classic hierarchy-based metric [50], [51] can be used to define the similarity of two concepts c_u and c_d, namely:

$$ConSim_1(c_u, c_d) = \frac{2 \cdot l(lca(c_u, c_d))}{l(c_u) + l(c_d)}, \tag{4}$$

where $lca(c_u, c_d)$ denotes the lowest common ancestor of the concepts c_u and c_d, and $l(c)$ denotes the level of a concept c (with $l(root) = 0$).

Moreover, since the shorter the shortest path between two concepts is, the (semantically) closer the two concepts are, another metric is to compute the similarity between two concepts by the reciprocal of their shortest path length [49]. Formally, for two concepts c_u and c_d:

$$ConSim_2(c_u, c_d) = \frac{1}{SP(c_u, c_d) + 1}, \tag{5}$$

where $SP(c_u, c_d)$ is the shortest path between c_u and c_d; and we have $SP(c_u, c_d) = l(c_u) + l(c_d) - 2 \cdot l(lca(c_u, c_d))$, if the ontology forms a tree structure, like the ODP taxonomy.

Example 2: Continuing Example 1, the concept similarities for the tags in $\vec{p}_u$ and $\vec{p}_d$ are computed as follows. For *Jaguar1*, we conduct breadth-first search for subtrees, using its ancestors (bottom-up) as subtree root, and find that its nearest concept is *Mammals*, and the *lca* is the *root* concept. As $l(root) = 0$ and $l(Jaguar1) = l(Mammals) = 3$, the concept similarity in Equation 4 is $ConSim_1(Jaguar1, Mammals) = \frac{2\times 0}{3+3} = 0$. Similarly, $ConSim_1(Apache, Mammals) = 0$. The concept similarity in Equation 5 for these two cases can be computed analogously. ■

2) ONTOLOGICAL SIMILARITY

Finally, the ontological similarity between two profiles $\vec{p}_u$ and $\vec{p}_d$ is computed as the weighted sum of all individual concept similarity values, which is an integration of the cosine or scalar similarity and the concept similarity. Formally, the cosine-based ontological similarity is defined as follows:

$$ontSim_{Cosine}(\vec{p}_u, \vec{p}_d) = \frac{\sum_{i=1}^{|T_u|}(w_{u_i} \cdot w_{d_i} \cdot ConSim(c_{u_i}, c_{d_i}))}{\sqrt{\sum_{i=1}^{|T_u|}(w_{u_i})^2} \cdot \sqrt{\sum_{j=1}^{|T_d|} k_j \cdot (w_{d_j})^2}}, \tag{6}$$

where $|T_u|$ and $|T_d|$ are the numbers of different tags in $\vec{p}_u$ and $\vec{p}_d$, respectively; c_{u_i} is the matching concept of a tag t_{u_i} in $\vec{p}_u$; c_{d_i} is c_{u_i}'s nearest concept mapped by t_{d_i} in $\vec{p}_d$; and w_{u_i}, w_{d_i}, and w_{d_j} are the weights of tags t_{u_i}, t_{d_i}, and t_{d_j}, respectively. As some concepts in $\vec{p}_d$ may be used multiple times as the nearest concept of different concepts in $\vec{p}_u$ (e.g., *Mammals* is used for both *Apache* and *Jaguar1* in Example 2), k_j is the number of times that t_{d_j}'s matching concept is used as the nearest concept, if t_{d_j} is a multi-selected tag; otherwise, $k_j = 1$ (see Lines 27–29 in Algorithm 3). $ConSim(c_{u_i}, c_{d_i})$ is the concept similarity of c_{u_i} and c_{d_i} as predefined, if t_{u_i} has a matching concept; if t_{u_i} is unmatched, then $ConSim(c_{u_i}, c_{d_i})$ is 1, if the same tag is in $\vec{p}_d$, and 0, otherwise.

Similarly, the scalar-based ontological similarity is formally defined as follows:

$$ontSim_{Scalar}(\vec{p}_u, \vec{p}_d) = \sum_{i=1}^{|T_u|}(w_{u_i} \cdot w_{d_i} \cdot ConSim(c_{u_i}, c_{d_i})), \tag{7}$$

The following example illustrates the computation of the above ontological similarity measure, as well as its advantages.

Example 3: Consider again the running example, and adopt Equation 4 for concept similarity. Then, it is straightforward to compute the cosine-based ontological similarity between $\vec{p}_u$ and $\vec{p}_d$ as $ontSim_{Cosine}(\vec{p}_u, \vec{p}_d) = 0$, because $ConSim_1(Jaguar1, Mammals) = ConSim_1(Apache, Mammals) = 0$.

However, given a new online document d' regarding a military organization with $\vec{p}_{d'} = \{(Military, 10)\}$, as in Examples 1 and 2, we obtain $ConSim_1(Jaguar1, Military) = ConSim_1(Apache, Military) = \frac{2\times 1}{1+3} = 0.5$. Subsequently, we compute the ontological similarity value between $\vec{p}_u$ and $\vec{p}_{d'}$ as follows:

$$ontSim_{Cosine}(\vec{p}_u, \vec{p}_{d'}) = \frac{5 \times 10 \times 0.5 + 3 \times 10 \times 0.5}{\sqrt{(5)^2 + (3)^2} \cdot \sqrt{2 \times (10)^2}} = 0.485,$$

where the weight of *Military* is multiplied by $k = 2$ in the normalization, because it is selected as the nearest concept twice for both *Jaguar1* and *Apache*.

Hence, if we only consider the personalization factor, d' will be re-ranked much higher than d, which is intuitively consistent with the user's preferences. On the contrary, if we adopt the cosine or scalar similarity for personalized re-ranking, it will result in an incorrect ranking: d is ranked higher than d', as $\vec{p}_d$ shares the tag *"Jaguar"* with $\vec{p}_u$. ■

V. PERSONALIZED RECOMMENDATION BASED ON ONTOLOGICAL SIMILARITY

We now present an algorithm using ontological similarity between the user profile and document profiles to achieve personalized recommendation on the Social Web.

Due to the relatively high computational cost of computing ontological similarities, in this work, to achieve a real-time online response of personalized search and recommendation, we reasonably assume that the user's preferences (described by the user profile) and the social summaries of documents (described by the document profiles) are usually stable in a short period of time. Therefore, for a given user, we first conduct an off-line pre-processing to compute the ontological similarities between this user's profile and the document profiles, and then store the resulting similarity values for the evaluation of online personalized recommendations. Note that, to adapt to less frequent changes of user and document profiles, periodic profile updates can be conducted, and the similarity computation process can be done again.

ALGORITHM 4: PrecomputeOntologicalSimilarity ($\vec{p}_u$, $list_{\vec{p}_d}$, Ont)

```
Input: User profile $\vec{p}_u$, a list of document profiles $list_{\vec{p}_d}$, and
       domain ontology $Ont$
Output: $list_{ontSim}$ — a list stores the resulting ontological
        similarity values
1 $hashmap_{\vec{p}_u} = TagAllocation(\vec{p}_u, Ont)$;            // Algorithm 1
2 foreach $\vec{p}_{d_i} \in list_{\vec{p}_d}$ do
3     $hashmap_{\vec{p}_d} = TagAllocation(\vec{p}_{d_i}, Ont)$;     // Algorithm 1
4     $Sim_{\vec{p}_{d_i}} = OntologicalSimilarity(\vec{p}_u, \vec{p}_{d_i}, hashmap_{\vec{p}_u},$
5     $hashmap_{\vec{p}_d}, Ont)$;                                   // Algorithm 3
6     $list_{ontSim}.add(Sim_{\vec{p}_{d_i}})$;
7 return $list_{ontSim}$.
```

The pre-computation of ontological similarities is described in Algorithm 4. It takes as *input* a weighted vector denoting a user u's profile ($\vec{p}_u$), a list of weighted vectors denoting document profiles ($list_{\vec{p}_d}$), and an external domain ontology (Ont). This algorithm consists of three steps: first, it allocates (maps) tags in $\vec{p}_u$ to concepts in Ont (Line 1); then, for each document d_i in the list $list_{\vec{p}_d}$, it allocates tags in $\vec{p}_{d_i}$ to concepts in Ont, computes the ontological similarity of $\vec{p}_u$ and $\vec{p}_{d_i}$, and includes the resulting ontological similarity in a list $list_{ontSim}$ (Lines 2–5). The list is finally returned for future use in personalized recommendation (Line 6).

The following result states the computational cost of Algorithm 4.

Proposition 3: Given a taxonomy Ont with r concepts, a user profile $\vec{p}_u$ of size m_u, and a document profile list $list_{\vec{p}_d}$ of length n, with document profiles $\vec{p}_{d_i}$ of size m_d, Algorithm 4 runs in time $O(n \cdot m_d \cdot r^2 + n \cdot m_u \cdot (r + m_d) + m_u \cdot r^2)$ in the worst case. If the underlying taxonomy is a balanced tree, then the worst-case complexity of Algorithm 4 is $O(n \cdot m_d \cdot r \cdot \log r + n \cdot m_u \cdot (r + m_d) + m_u \cdot r \cdot \log r)$.

Proof: According to Proposition 1, allocating user profile tags in Line 1 takes $O(m_u \cdot r^2)$ (resp., $O(m_u \cdot r \cdot \log r)$ for a balanced-tree taxonomy) in the worst case. In addition, the for loop in Line 2 takes n steps, each of which in the worst case needs $O(m_d \cdot r^2 + m_u \cdot (r + m_d))$ (resp., $O(m_d \cdot r \cdot \log r + m_u \cdot (r + m_d))$ for the balanced-tree case): specifically, allocating document profile tags in Line 3 takes $O(m_d \cdot r^2)$ (resp., $O(m_d \cdot r \cdot \log r)$), and the computation of the ontological similarity in Line 4 takes $O(m_u \cdot (r + m_d))$ based on Proposition 2. So, the total worst-case running time of Algorithm 4 is $O(n \cdot m_d \cdot r^2 + n \cdot m_u \cdot (r + m_d) + m_u \cdot r^2)$ (resp., $O(n \cdot m_d \cdot r \cdot \log r + n \cdot m_u \cdot (r + m_d) + m_u \cdot r \cdot \log r)$ for the balanced-tree case). □

So, based on Equation 1, personalized recommendation can be done by sorting the documents according to ontological similarity values in the returned list. The next result states the computational cost of this ontological-similarity-based recommendation.

Proposition 4: Given a pre-computed ontological similarity list $list_{ontSim}$ of size n, the time needed for personalized recommendation using ontological similarity is $O(n \cdot \log n)$ in the worst case.

Proof: Given the ontological similarity values, for personalized recommendation we have to sort n documents, which is possible in time $O(n \cdot \log n)$ in the worst case. □

VI. EXPERIMENTAL STUDIES

In this section, extensive experiments are conducted to evaluate the performance of the ontological-similarity-based personalized recommender systems from three perspectives: content-based personalized recommendation, tag disambiguation, and tag-to-ontology allocation.

Experiments are conducted over a public real-world *Delicious* dataset, which is gathered from the Delicious bookmarking system, released in HetRec 2011 [65], and used in several existing tag-aware personalized recommender systems [21], [28]. For a fair comparison, the same preprocessing as in [21], [28] is conducted to remove infrequent tags that are used less than 15 times. The statistical information about the resulting dataset is as shown in Table 1, where 41.9% of users assigned less than 50 tags; 31.3% and 22.8% of users had 50-100 and 100-200 tags, respectively; and only 4% of users have more than 200 tags.

TABLE 1. Dataset information.

Users	Tags	Documents	Assignments
1 843	3 508	65 877	339 744

Furthermore, as stated previously, we use the ODP taxonomy as the underlying ontology, which contains 802, 456 categories (i.e., concepts) in total. Note also that 1, 637 tags (47.7% of all the tags) in the Delicious dataset can be mapped to the ODP taxonomy. All models are implemented using Python, and run on a server with two 8-core Haswell processors and 128 GB memory.

A. MAIN PERFORMANCE IN CONTENT-BASED PERSONALIZED RECOMMENDATION

In this section, we evaluate the performance of the proposed ontological-similarity-based recommendation model in content-based tag-aware personalized recommendation. Here, we implement the ontological-similarity-based recommender systems using all combinations of concept similarity measures and ontological similarity measures introduced in this chapter, resulting in two systems based on scalar-based ontological similarity using either concept similarity $ConSim_1$ (denoted $OntoScalar_1$) or $ConSim_2$ (denoted $OntoScalar_2$), and the other two systems based on the cosine-based ontological similarities (denoted $OntoCos_1$ and $OntoCos_2$, respectively).

1) EVALUATION METHODOLOGY

Although the relevance and/or value of personalization depends on users' subjective views, several studies [66]–[68] have shown that users' tagging behavior is closely correlated with their personal relevance judgment; i.e., if a document is annotated by a user with certain tags, this document is very likely to be visited by the same user if it appears as a recommendation result. This is the basis of our automatic evaluation: the plausible documents in personalized recommendation are the ones annotated by a user.

Therefore, we assume that the target documents of a given user are those annotated by this user. Thus, for each user u in the dataset, we randomly select 1 to 3 documents d annotated by this user as his/her target document and put the selected (u, d) pairs in a testing dataset, where we aim at recommending d given u in the testing. Please note that the reason of selecting only a few (1 to 3) user-item testing samples for each user is to better simulate the practical usage of lightweight recommender systems. Then, for each (u, d) pair in the testing set, we remove from the dataset all the assignments given by u to d; the remaining assignments in the dataset are used as a historical dataset, which is used to construct user and item profiles for content-based tag-aware personalized recommendations and also used for training in machine-learning-based baselines. Finally, we get 3, 529 user-item pairs in the testing dataset and 274, 362 assignments in the history dataset.

As for the evaluation of the personalized recommendation, the most popular metrics are precision, recall, and F1-score [24]. Since most users usually only browse the topmost recommended news, we apply these metrics at a given cut-off rank k, i.e., considering only the top-k results on the recommendation list, called *precision at k* ($P@k$), *recall at k* ($R@k$), and *F1-score at k* ($F@k$). Formally,

$$P@k = \frac{1}{|U'|}\sum\nolimits_{u\in U'} P_u@k, \quad P_u@k = \frac{C_u@k}{k}, \tag{8}$$

$$R@k = \frac{1}{|U'|}\sum\nolimits_{u\in U'} R_u@k, \quad R_u@k = \frac{C_u@k}{N_u}, \tag{9}$$

$$F@k = \frac{2\cdot P@k\cdot R@k}{P@k+R@k}, \tag{10}$$

where k is the length of the recommendation list, U' is the set of users in the test (sub)sets, $P_u@k$ and $R_u@k$ are the precision and recall at k for a given user u, respectively, $C_u@k$ is the number of u's target news in the recommendation list, and N_u is the total number of u's target news in test (sub)sets.

Although precision, recall, and F1-score are very useful metrics, they do not consider the positions of the target documents in a recommendation list. In practice, users always prefer to have their target documents ranked as top as possible in a recommendation list; therefore, we also employ the *mean reciprocal rank (MRR)* [69] as evaluation metric, which gives greater importance to documents ranked higher. MRR measures the performance of a personalized function by assigning a value $1/r$ for each test sample and then computing the mean value. Formally,

$$MRR = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{r_j}, \tag{11}$$

where r_j is the ranking position of the target document of the j-th test sample in the personalized recommendation list, and n is the total number of test samples.

2) EVALUATION VS. CONVENTIONAL SIMILARITY SOLUTIONS

We first compare the personalized recommendation performance of tag-aware content-based recommender systems using ontological similarities to those using the conventional cosine and scalar similarities. Therefore, two content-based tag-aware personalized recommender systems based on the widely used state-of-the-art similarity metrics, cosine similarity [15] and scalar similarity [20], are selected as baselines.

Figure 4 depicts in detail the personalized recommendation performance of the four ontological-similarity-based recommender systems (denoted $OntScalar_1$, $OntScalar_2$, $OntCos_1$, and $OntCos_2$) and the cosine-similarity-based and scalar-similarity-based recommender systems (denoted *Cosine* and *Scalar*, respectively) on the Delicious dataset, in terms of MRR, $P@k$, $R@k$, and $F@k$ with cut-off ranks $k = 5, 10, \ldots, 40$. In addition, to evaluate their general performance, we further calculate the average $P@k$, $R@k$, and $F@k$ of these recommender systems, which are then summarized, together with their numerical values of MRR, in Table 2.

Generally, as shown in Figure 4 and Table 2, the recommender systems based on the scalar-based similarity metrics, *Scalar*, $OntScalar_1$, and $OntScalar_2$, greatly outperform those based on the cosine-based similarity metrics, *Cosine*, $OntCos_1$, and $OntCos_2$. This is consistent with the results reported in [20] that scalar similarity can outperform cosine similarity in content-based tag-aware personalized recommendation by eliminating the profile length normalization factors. In addition, the recommender systems based on the scalar-based ontological similarities, $OntScalar_1$ and $OntScalar_2$, always outperform the recommender system based on the traditional scalar similarity, *Scalar*, at all cut-off ranks k and in terms of all evaluation metrics. Similarly, the recommender systems using the cosine-based ontological similarities, $OntCos_1$ and $OntCos_2$, also outperform the one using the cosine similarity, *Cosine*, in all cases. This finding demonstrates our conclusion that the proposed ontological similarity can significantly improve the performance of traditional similarity metrics in content-based tag-aware personalized recommendation by using ontologies to address the tag ambiguity and redundancy problems, and to semantically measure the profile similarities.

Finally, the recommender systems using ontological similarities that are based on $Consim_1$, i.e., $OntScalar_1$

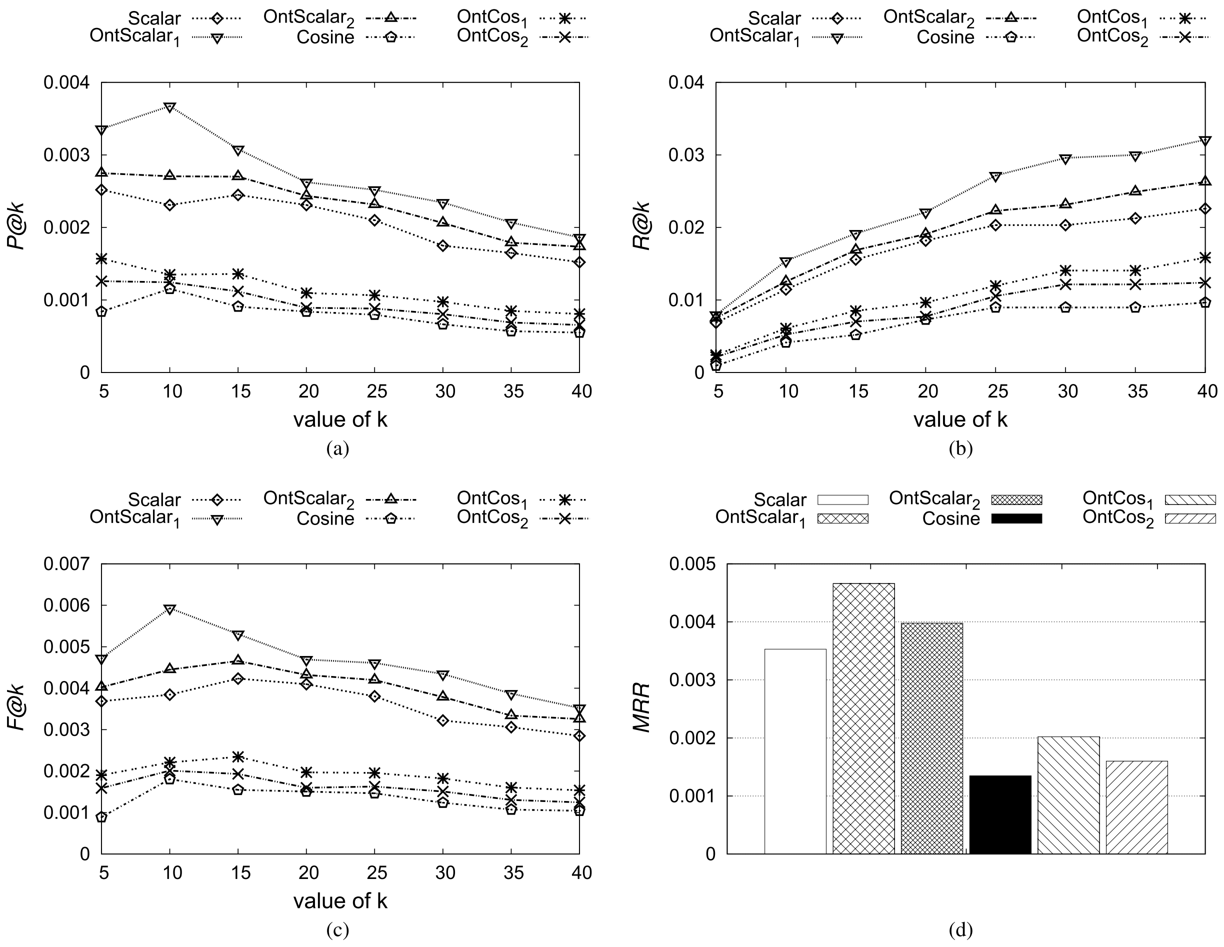

FIGURE 4. The recommendation performance of *Scalar*, *OntScalar*$_1$, *OntScalar*$_2$, *Cosine*, *OntCosine*$_1$, and *OntScalar*$_2$. (a) Precision at *k*. (b) Recall at *k*. (c) F1-score at *k*. (d) Mean Average Precision at *k*.

TABLE 2. Average *P@k*, *R@k*, *F@k*, and MRR of *Scalar*, *OntScalar*$_1$, *OntScalar*$_2$, *Cosine*, *OntCos*$_1$, and *OntCos*$_2$ when $k = 5, 10, \ldots, 40$.

	P@k	R@k	F@k	MRR
Scalar	0.00207	0.01708	0.00360	0.00353
OntScalar$_1$	**0.00268**	**0.02293**	**0.00462**	**0.00466**
OntScalar$_2$	0.00231	0.01907	0.00400	0.00398
Cosine	0.00079	0.00676	0.00132	0.00135
OntCos$_1$	**0.00113**	**0.01032**	**0.00191**	**0.00202**
OntCos$_2$	0.00094	0.00867	0.00160	0.00160

and *OntCos*$_1$, generally outperforms those using *Consim*$_2$ based ontological similarities, i.e., *OntScalar*$_2$ and *OntCos*$_2$. Specifically, *OntScalar*$_1$ achieves the best performance among scalar-based recommender systems: its performance is 16.0%, 20.2%, 15.5%, and 17.1% (resp., 29.5%, 34.3%, 28.3%, and 32.0%) higher than that of *OntScalar*$_2$ (resp., *Scalar*) in terms of the average $P@k$, $R@k$, $F@k$, and MRR, respectively. Similarly, the performance of *OntCos*$_1$ is 20.2%, 19.0%, 19.4%, and 26.3% (resp., 43.0%, 52.7%, 44.7%, and 49.6%) higher than that of *OntCos*$_2$ (resp., *Cosine*) in the average $P@k$, $R@k$, $F@k$, and MRR, respectively. This observation asserts that the concept similarity *Consim*$_1$, that is based on relative concept depth (as defined in Equation 4), may be more effective than the shortest-path-based concept similarity *Consim*$_2$ (as defined in Equation 5) in measuring the semantic relevance between matching concepts of tags in profiles, and thus achieves better performance in ontology-based personalized recommendation.

3) EVALUATION VS. MACHINE-LEARNING-BASED SOLUTIONS

We further evaluate the personalized recommendation performance of the ontological-similarity-based recommender

TABLE 3. OntScalar$_1$ vs. Clustering, Autoencoder, and DSPR.

	avg. time	$P@k$	$R@k$	$F@k$	MRR
Clustering	1.26 secs	0.00232	0.02076	0.00392	0.00398
Autoencoder	1.45 secs	0.00243	0.02152	0.00417	0.00421
DSPR	20.3 secs	**0.003315**	**0.03147**	**0.00625**	**0.00645**
$OntScalar_1$	**0.17 secs**	0.00268	0.02293	0.00462	0.00466

systems to those using machine-learning-based solutions. To this end, we use three state-of-the-art machine-learning-based solutions for the tag ambiguity problem (i.e., clustering [19], autoencoders [21], and DSPR [28]) as baselines: (i) Clustering-based solution: hierarchical clustering [19] is applied to model the users and documents as cluster-based feature vectors, upon which content-based filtering using scalar similarity is applied for recommendations. (ii) Autoencoder-based solution: an autoencoder [21] is used to obtain abstract representations of user and document profiles, upon which content-based filtering using scalar similarity is applied for recommendations. (iii) Deep-semantic similarity-based personalized recommendation (DSPR) [28]: two neural networks are applied to learn abstract feature representations of user and document profiles, which are trained to maximize (resp., minimize) the similarities between users and their target items (resp., irrelevant items); then, the learned deep-semantic similarity was directly used to generate recommendations. The model parameters of hierarchical clustering, autoencoder, and DSPR follow the settings in [19], [21], and [28], respectively.

Table 3 shows the content-based tag-aware recommendation performance of the best ontological-similarity-based recommender system, $OntScalar_1$, and the state-of-the-art machine-learning-based tag-aware recommender systems using clustering, autoencoder, and DSPR in terms of average time-cost of a testing case, $P@k$, $R@k$, $F@k$, and MRR.

The comparative results of clustering, autoencoder, and DSPR in Table 3 is consistent with the results reported in [28], and we generally have the following observations from Table 3. (i) The recommender system based on scalar-based ontological similarity has much lower time-cost for recommendation than the machine-learning-based recommender systems: the recommendation processing of OntScalar$_1$ is about 6.4, 7.5, and 118.4 times quicker than those of clustering, autoencoder, and DSPR models, respectively. This observation supports our argument that, due to the need of model training, the machine-learning-based solutions are usually much more computationally expensive than the proposed ontological-similarity-based solution. (ii) The recommendation accuracy of $OntScalar_1$ is better than those of clustering and autoencoders, but is worse than the one of DSPR, in terms of $P@k$, $R@k$, $F@k$, and MRR. This is because, by using recommendation-oriented learning objective, DSPR can learn very effective latent feature representations for personalized recommendation, while the proposed ontology-based solution only relies on human-input feature information, which is sometimes not comprehensive enough to achieve the most effective feature representations. However, please also note that although the recommendation accuracy of $OntScalar_1$ is lower than DSPR, its recommendation processing is 118.4 times quicker than the one of DSPR. Therefore, we can assert that the proposed ontology-based solution may not be the best choice, if large datasets and powerful computational facilities are easily accessible; however, due to its low computational cost and reasonable recommendation accuracy, the proposed ontology-based solution is a good disambiguation choice for lightweight recommender systems and a complement to machine-learning-based recommendation solutions.

B. PERFORMANCE IN TAG DISAMBIGUATION

1) BASELINES

As discussed in Section II, there are existing works also focusing on the similarity between tags, which can be applied to disambiguate synonyms and homonyms using their most related tags. According to the results in [26] and [27], distributional measures generally have the best performance in measuring tag relatedness. Therefore, to show the strength of the proposed ontology-based solution in solving the tag ambiguity problem, tag-context-based, document-context-based, and user-context-based distributional measures (denoted *tagCont*, *docCont*, and *userCont*, respectively) as defined in [26] are selected as the baselines of the tag disambiguation performance evaluation.

2) EVALUATION METHODOLOGY

As distributional measures rely on the most related tags for tag disambiguation, their tag disambiguation performance depends on the semantic distance in the ontology between tags and their most related tags. Similarly, as for the proposed ontology-based solution, the most related tag for a tag can be defined as the tag whose mapping concept has the highest concept similarity with the matching concept of the given tag. Consequently, we use the *average semantic distance* between tags and their most related tags to evaluate the tag disambiguation performance of the proposed ontology-based solution and the three baselines. For a fair comparison, the semantic distance is estimated by the *shortest-path-based semantic grounding* measurement as used in [26] and [27]; here, ODP is used as the underlying ontology for semantic grounding, and cosine similarity is used to estimate the similarity between distributional vectors for the baselines.

The shortest-path-based semantic grounding is conducted using the same process as in [26]: for each tag in the Delicious dataset, we first find its most related tags using the distributional measures tagCont, docCont, and userCont, as well as the ontology-based solution. Then, for each tag with matching concept in the ODP taxonomy, if its most related tag also has matching concepts, we estimate the semantic distance between the matching concepts of this tag and its most related tag by the length of the shortest path between the concepts. If either the original or the most related tag has more than one matching concept, we compute the semantic distance for all possible combinations of matching concepts and select the minimal value. As for the proposed ontology-based solution, both concept measures based on the relative depth of concepts and the shortest path (as defined in Equation (4) and (5)) are used and denoted as *Onto-based*$_1$ and *Onto-based*$_2$, respectively.

3) RESULTS

Figure 5 depicts the average semantic distance in terms of the average length of the shortest paths in the ODP taxonomy from the matching concepts of original tags to those of the most related tags generated by the *docCont*, *tagCont*, *userCont*, and *Onto-based* methods. Generally, the shorter the average shortest path, the closer the semantics of the most related tags are to the ordinal tags and the better disambiguation performance in synonyms and homonyms the solution will have.

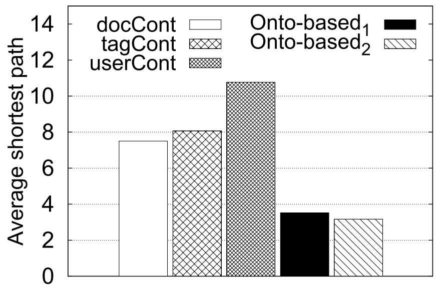

FIGURE 5. Average semantic distance based on the average length of the shortest paths between the matching concepts of original tags and those of their most related tags in the ODP taxonomy generated by the *docCont*, *tagCont*, *userCont*, and *Onto-based* solutions.

As shown in Figure 5, both *Onto-based*$_1$ and *Onto-based*$_2$ greatly outperform all the baselines: their average shortest path is less than half of that of the best baseline, *docCont*, while the performance of *Onto-based*$_2$ is slightly better than *Onto-based*$_1$. The superior performance of ontology-based methods is mainly because of the following reason: for a given tag, the ontology-based methods find their most related tag as the tag mapping to the concept with highest concept similarity to the original tag's matching concept in ODP; especially, for *Onto-based*$_2$, the matching concept of the most related tag found by *Onto-based*$_2$ is guaranteed to always have the best shortest path to the original tag's matching concept among all matching concepts in ODP; so, its average shortest path is guaranteed to be optimal.

Despite achieving superior performance in tag disambiguation, the ontology-based solution has still some limitations: due to the need of mapping tags to ontologies, only the ambiguity in tags that can be mapped to ontologies are handled. Although we have considered the similarity between unmatched tags using the cosine similarity (Lines 20-25 in Algorithm 1), it will still be beneficial to integrate the non-ontology-based disambiguation solutions, such as co-occurrence tags and distributional measures [27], to further address the ambiguity problem in the unmatched tags, so as to further enhance the recommendation performance. As we focus on an ontology-based solution in this work, this hybrid solution will be investigated in future work.

C. PERFORMANCE IN TAG-TO-ONTOLOGY MAPPING

1) BASELINE

As for allocating tags to the underlying ontology, the main challenge is the multiple occurrence of matching concepts due to homonyms. The state-of-the-art tag allocation strategy [57] first maps a given tag to all possible candidate concepts in the ontology and then uses the contextual information of its profile to identify the most relevant concept by computing the similarity of all combinations of the given tag and other tags in this profile using the Wu and Palmer similarity [50] in the ontology. This strategy is denoted *Wu & Palmer* and considered as the competitor to the proposed top-down traversal tag allocation strategy in evaluating its tag allocation performance.

2) EVALUATION METHODOLOGY

To evaluate the tag-to-ontology allocation accuracy, we investigate how semantically relevant the matching concept selected by a given tag allocation strategy is to the profile context of the given tag, such that the more the selected matching concept is semantically relevant to the profile context, the better the tag allocation strategy.

Therefore, we design an automatic quantitative evaluation based on the following assumption: for a given tag in a given profile, its selected matching concept is believed to be semantically relevant to the context of the profile, if this matching concept's average semantic distance to the matching concepts of all other tags in this profile is low; so, the lower the average semantic distance, the more semantically relevant the matching concept to the profile context.

Therefore, for each tag in a given user or document profile having a matching concept selected by a tag allocation strategy, we compute the average semantic distance between its matching concept to all other selected matching concepts in the same profile. Then, the tag-to-ontology allocation performance of the two strategies are evaluated based on a *mean average semantic distance* (denoted *MASD*), formally defined as follows:

$$MASD = \frac{1}{N_p}\sum_{a=1}^{N_p}\frac{1}{N_a^t}\sum_{j=1}^{N_a^t} aveSemDis(c_{a,j}), \quad (12)$$

where N_p is the number of profiles, N_a^t is the number of tags with matching concepts in profile a, $c_{a,j}$ is the selected matching concept of tag j in profile a, and $aveSemDis(c_{a,j})$ is the average semantic distance of $c_{a,j}$ to all other selected matching concepts in profile a. Here, as in Section VI-B, the shortest path is used as the measure of semantic distance.

In addition, as the first step for computing the ontological similarity, tag allocation is invoked very frequently in ontology-based recommendation; therefore, besides allocation accuracy, the allocation efficiency is also a very critical factor to ensure the quick response for real-time online recommendation. Actually, as stated in Section IV-A, the purpose of proposing the top-down traversal tag allocation strategy in this work is to overcome the high computational complexity problem in the existing strategy [57]. Therefore, we record the average tag allocation time-cost of both strategies, where the top-down traversal strategy is expected to have much shorter allocation time than the baseline.

3) EXPERIMENTAL RESULTS

Table 4 shows the mean average semantic distance (MASD) of allocating tags in all document and user profiles (denoted $MASD_d$ and $MASD_u$, respectively), using the existing strategy based on Wu and Palmer similarity (denoted *Wu & Palmer*) and the proposed top-down traversal strategy (denoted *Top-Down*), as well as their average time-costs for allocating tags for each user and document profile (denoted $time_d$ and $time_u$, respectively).

TABLE 4. Accuracy and efficiency of tag-to-ontology allocation.

	$MASD_d$	$time_d$	$MASD_u$	$time_u$
Wu & Palmer	3.708	0.280 secs	8.785	2.69 secs
Top-Down	4.069	0.057 secs	9.415	0.44 secs

As shown in Table 4, for tag allocation in document profiles, the $MASD_d$'s of *Wu & Palmer* and *Top-Down* are very close (3.708 vs. 4.069), while the time needed for *Wu & Palmer* is 4.9 times as long as the one of *Top-Down* (0.280 vs. 0.057 seconds). Similarly, for the user profiles, the $MASD_u$ of *Top-Down* is only 7.1% higher than the one of *Wu & Palmer* (8.785 vs. 9.415), but its mapping efficiency is about 5.1 times quicker than the one of *Wu & Palmer* (2.69 vs. 0.44 seconds). These results prove that the proposed top-down traversal tag allocation strategy is much more efficient (around 5 times quicker) than the existing tag allocation baseline [57], while maintaining a very close allocation accuracy. This is also consistent with our computational complexity analysis in Section IV-A.

VII. SUMMARY AND OUTLOOK

In this paper, we have proposed an effective ontological similarity measure that uses ontologies to solve the tag ambiguity problem and to semantically measure the similarity between user and document profiles. More precisely, its novelty lies in its use of concept similarity in ontologies to estimate the semantic relevance of tags in profiles. We have first developed a two-step top-down disambiguation algorithm in order to solve the multiple occurrence challenge at the stage of tag allocation. Then, we have proposed an algorithm using the concept similarity in the ontology to compute the ontological similarity. Finally, we have also presented methods of personalized recommendation on the Social Web using ontological similarity. A complexity analysis of these algorithms has shown that the approach of using ontological similarity for personalization is practically tractable. In addition, extensive experimental studies based on a public real-world dataset have shown that: (i) the proposed ontological-similarity-based recommender systems are more effective than the state-of-the-art cosine-similarity-based and scalar-similarity-based recommender systems in content-based tag-aware personalized recommendations in terms of all evaluation metrics; (ii) the recommendation accuracy of our ontological-similarity-based solution is better than those of the clustering and autoencoder baselines, but is worse than that of DSPR-based baseline; however, its computational cost is much lower than those of all machine-learning-based lines (in particular, it is 118.4 times quicker than DSPR); (iii) the proposed ontology-based solution greatly outperforms (with more than double the performance) the state-of-the art baselines in tag disambiguation; and (iv) the proposed top-down traversal tag allocation strategy is much more efficient (around five times quicker) than the existing tag allocation strategy, while maintaining a similar allocation accuracy.

A. ADVANTAGES AND DISADVANTAGES

The main advantage of our approach is that no training is required for personalized recommendation; the only requirement is that we have a well-defined ontology or taxonomy. This is very useful in cases in which we do not have resources for training the model, such as in mobile apps, or if the data are so fast-changing that re-training is needed very often and becoming very expensive.

Another advantage of our algorithms is that the recommendation results of this work are much more transparent and explainable than the ones provided by machine-learning approaches, especially those that use deep learning. Consequently, users are likely to be more willing to trust the results of our systems, because they tend to be easier to understand.

The main drawback in our work is that it only relies on human input feature information, which is sometimes not comprehensive enough to achieve the most effective feature representations. Therefore, the recommendation accuracy of our ontology-based solution is sometimes lower than some of the machine-learning based solutions, which are capable of discovering the latent features of users to provide sufficient and comprehensive feature information. The proposed ontology-based solution may thus not be the best choice, if large datasets and powerful computational facilities are easily accessible; however, it is a good disambiguation choice for lightweight recommender systems and a complement to machine-learning-based recommendation solutions, because of its low computational cost and reasonable recommendation accuracy.

B. FUTURE WORK

In the future, we will conduct further experiments involving other Social Web datasets, different ontologies, and other concept similarity metrics to investigate the change of effectiveness of ontological similarity for various social Web resources and ontologies. In addition, another topic for future research is to further orthogonally consider the tag relatedness measures (e.g., co-occurrence tags and distributional measures [27]) as solutions to further address the ambiguity problem for the tags that cannot be matched to ontologies. Finally, apart from content-based solutions, we will also work on improving collaborative filtering-based (e.g., matrix factorization) and graph-based (e.g., FolkRank) solutions in folksonomy-based personalized recommendation.

ACKNOWLEDGMENTS

The authors acknowledge the use of Oxford University's ARC resources.

REFERENCES

[1] M. R. Bouadjenek, H. Hacid, and M. Bouzeghoub, "Social networks and information retrieval, how are they converging? A survey, a taxonomy and an analysis of social information retrieval approaches and platforms," *Inf. Syst.*, vol. 56, pp. 1–18, Mar. 2016.
[2] A. Capocci and G. Caldarelli, "Folksonomies and clustering in the collaborative system *CiteULike*," *J. Phys. A, Math. Theor.*, vol. 41, no. 22, p. 224016, 2008.
[3] F. Gedikli and D. Jannach, "Rating items by rating tags," in *Proc. ACM Conf. Recommender Syst. (RecSys)*, 2010, pp. 25–32.
[4] N. Landia and S. Anand, "Personalised tag recommendation," in *Proc. ACM Conf. Recommender Syst. (RecSys)*, 2009, pp. 83–86.
[5] X. Shen, B. Tan, and C. Zhai, "Implicit user modeling for personalized search," in *Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2005, pp. 824–831.
[6] A. Micarelli and F. Sciarrone, "Anatomy and empirical evaluation of an adaptive Web-based information filtering system," *User Model. User-Adapted Interaction*, vol. 14, nos. 2–3, pp. 159–200, 2004.
[7] T. Lappas, K. Liu, and E. Terzi, "Finding a team of experts in social networks," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2009, pp. 467–476.
[8] X. Shen and C. X. Zhai, "Exploiting query history for document ranking in interactive information retrieval," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2003, pp. 377–378.
[9] F. Silvestri, "Mining query logs: Turning search usage data into knowledge," *Found. Trends Inf. Retr.*, vol. 4, nos. 1–2, pp. 1–174, 2010.
[10] M. Speretta and S. Gauch, "Personalized search based on user search histories," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Sep. 2005, pp. 622–628.
[11] K. Sugiyama, K. Hatano, and M. Yoshikawa, "Adaptive Web search based on user profile constructed without any effort from users," in *Proc. Int. World Wide Web Conf. (WWW)*, 2004, pp. 675–684.
[12] J. Luxenburger, S. Elbassuoni, and G. Weikum, "Task-aware search personalization," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2008, pp. 721–722.
[13] J. Teevan, S. T. Dumais, and D. J. Liebling, "To personalize or not to personalize: Modeling queries with variation in user intent," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2008, pp. 163–170.
[14] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay, "Accurately interpreting clickthrough data as implicit feedback," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2005, pp. 154–161.
[15] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu, "Exploring folksonomy for personalized search," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2008, pp. 155–162.
[16] Z.-K. Zhang, T. Zhou, and Y.-C. Zhang, "Tag-aware recommender systems: A state-of-the-art survey," *J. Comput. Sci. Technol.*, vol. 26, no. 5, pp. 767–777, Sep. 2011.
[17] M. R. Bouadjenek, H. Hacid, and M. Bouzeghoub, "Sopra: A new social personalized ranking function for improving Web search," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2013, pp. 861–864.
[18] I. Cantador, A. Bellogín, and D. Vallet, "Content-based recommendation in social tagging systems," in *Proc. ACM Conf. Recommender Syst. (RecSys)*, 2010, pp. 237–240.
[19] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke, "Personalized recommendation in social tagging systems using hierarchical clustering," in *Proc. ACM Conf. Recommender Syst. (RecSys)*, 2008, pp. 259–266.
[20] D. Vallet, I. Cantador, and J. M. Jose, "Personalizing Web search with folksonomy-based user and document profiles," in *Proc. Eur. Conf. Inf. Retr. (ECIR)*, 2010, pp. 420–431.
[21] Y. Zuo, J. Zeng, M. Gong, and L. Jiao, "Tag-aware recommender systems based on deep neural networks," *Neurocomputing*, vol. 204, pp. 51–60, Sep. 2016.
[22] M. R. Bouadjenek, H. Hacid, M. Bouzeghoub, and A. Vakali, "Using social annotations to enhance document representation for personalized search," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2013, pp. 1049–1052.
[23] K. H. L. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme, "Tag-aware recommender systems by fusion of collaborative filtering algorithms," in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2008, pp. 1995–1999.
[24] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowl.-Based Syst.*, vol. 46, pp. 109–132, Jul. 2013.
[25] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.
[26] C. Cattuto, D. Benz, A. Hotho, and G. Stumme, "Semantic grounding of tag relatedness in social bookmarking systems," in *Proc. Int. Semantic Web Conf. (ISWC)*, 2008, pp. 615–631.
[27] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme, "Evaluating similarity measures for emergent semantics of social tagging," in *Proc. Int. World Wide Web Conf. (WWW)*, 2009, pp. 641–650.
[28] Z. Xu, C. Chen, T. Lukasiewicz, Y. Miao, and X. Meng, "Tag-aware personalized recommendation using a deep-semantic similarity model with negative sampling," in *Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2016, pp. 1921–1924.
[29] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *Int. J. Hum.-Comput. Stud.*, vol. 43, nos. 5–6, pp. 907–928, 1995.
[30] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, 1986.
[31] P.-A. Chirita, C. S. Firan, and W. Nejdl, "Personalized query expansion for the Web," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2007, pp. 7–14.
[32] A. Kobsa, "Privacy-enhanced personalization," *Commun. ACM*, vol. 50, no. 8, pp. 24–33, 2007.
[33] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme, "Information retrieval in folksonomies: Search and ranking," in *Proc. Eur. Semantic Web Conf. (ESWC)*, 2006, pp. 411–426.
[34] S. Maniu and B. Cautis, "Network-aware search in social tagging applications: Instance optimality versus efficiency," in *Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2013, pp. 939–948.
[35] B. Kahveci, I. S. Altingövde, and Ö. Ulusoy, "Integrating social features into mobile local search," *J. Syst. Softw.*, vol. 122, pp. 155–164, Dec. 2016.
[36] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su, "Optimizing Web search using social annotations," in *Proc. Int. World Wide Web Conf. (WWW)*, 2007, pp. 501–510.

[37] Y. A. Kim and G. W. Park, "Topic-driven socialrank: Personalized search result ranking by identifying similar, credible users in a social network," *Knowl.-Based Syst.*, vol. 54, pp. 230–242, Dec. 2013.
[38] H. Xie *et al.*, "Incorporating sentiment into tag-based user profiles and resource profiles for personalized search in folksonomy," *Inf. Process. Manage.*, vol. 52, no. 1, pp. 61–72, 2016.
[39] Z. Saoud, S. Kechid, M. Saoud, and A. Doucet, "Exploiting social annotations to generate resource descriptions in a distributed environment: Cooperative multi-agent simulation on query-based sampling," *Rev. Socionetw. Strategies*, vol. 11, no. 1, pp. 83–93, 2017.
[40] J. Wei and F. Meng, "Personalized information recommendation based on synonymy tag optimization," *Cluster Comput.*, pp. 1–12, 2017.
[41] M. R. Bouadjenek, H. Hacid, M. Bouzeghoub, and A. Vakali, "PerSaDoR: Personalized social document representation for improving Web search," *Inf. Sci.*, vol. 369, pp. 614–633, Nov. 2016.
[42] M. G. Campana and F. Delmastro, "Recommender systems for online and mobile social networks: A survey," *Online Soc. Netw. Media*, vol. 3, pp. 75–97, Oct. 2017.
[43] A. Klašnja-Milićević, B. Vesin, M. Ivanović, Z. Budimac, and L. C. Jain, "Folksonomy and tag-based recommender systems in e-learning environments," in *E-Learning Systems*, 2017, pp. 77–112.
[44] K. Haruna *et al.*, "Context-aware recommender system: A review of recent developmental process and future research direction," *Appl. Sci.*, vol. 7, no. 12, p. 1211, 2017.
[45] F. Gasparetti, "Personalization and context-awareness in social local search: State-of-the-art and future research challenges," *Pervasive Mobile Comput.*, vol. 38, pp. 446–473, Jul. 2017.
[46] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[47] Y. Zhen, W.-J. Li, and D.-Y. Yeung, "TagiCoFi: Tag informed collaborative filtering," in *Proc. ACM Conf. Recommender Syst. (RecSys)*, 2009, pp. 69–76.
[48] L. Ding *et al.*, "Swoogle: A search and metadata engine for the semantic Web," in *Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2004, pp. 652–659.
[49] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 1, pp. 17–30, Jan. 1989.
[50] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 1994, pp. 133–138.
[51] P. Ganesan, H. Garcia-Molina, and J. Widom, "Exploiting hierarchical domain structure to compute similarity," *ACM Trans. Inf. Syst.*, vol. 21, no. 1, pp. 64–93, 2003.
[52] J. J. Jiang and D. W. Conrath. (2017). "Semantic similarity based on corpus statistics and lexical taxonomy." [Online]. Available: https://arxiv.org/abs/cmp-lg/9709008
[53] D. Lin, "An information-theoretic definition of similarity," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 1998, pp. 296–304.
[54] Y. Li, Z. A. Bandar, and D. Mclean, "An approach for measuring semantic similarity between words using multiple information sources," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 871–882, Jul. 2003.
[55] G. Hirst and D. St-Onge, "Lexical chains as representations of context for the detection and correction of malapropisms," in *WordNet: An Electronic Lexical Database*. Cambridge, MA, USA: MIT Press, 1998, pp. 305–332.
[56] X. Han, Z. Shen, C. Miao, and X. Luo, "Folksonomy-based ontological user interest profile modeling and its application in personalized search," in *Proc. Int. Conf. Active Media Technol. (AMT)*, 2010, pp. 34–46.
[57] S. Angeletou, M. Sabou, and E. Motta, "Semantically enriching folksonomies with FLOR," in *Proc. Extended Semantic Web Conf. (ESWC) Workshop Collective Intell. Semantic Web*, 2008, pp. 65–79.
[58] H. Movahedian and M. R. Khayyambashi, "Folksonomy-based user interest and disinterest profiling for improved recommendations: An ontological approach," *J. Inf. Sci.*, vol. 40, no. 5, pp. 594–610, 2014.
[59] I.-C. Hsu, "Integrating ontology technology with folksonomies for personalized social tag recommendation," *Appl. Soft Comput.*, vol. 13, no. 8, pp. 3745–3750, 2013.
[60] L. Ratinov, D. Roth, D. Downey, and M. Anderson, "Local and global algorithms for disambiguation to Wikipedia," in *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2011, pp. 1375–1384.
[61] C. Li, A. Sun, and A. Datta, "A generalized method for word sense disambiguation based on Wikipedia," in *Proc. Eur. Conf. Inf. Retr. (ECIR)*, 2011, pp. 653–664.
[62] C. Li, A. Sun, and A. Datta, "TSDW: Two-stage word sense disambiguation using Wikipedia," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 64, no. 6, pp. 1203–1223, 2013.
[63] D. Milne and I. H. Witten, "An effective, low-cost measure of semantic relatedness obtained from Wikipedia links," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2008, pp. 25–30.
[64] E. G. M. Petrakis, G. Varelas, A. Hliaoutakis, and P. Raftopoulou, "X-Similarity: Computing semantic similarity between concepts from different ontologies," *J. Digit. Inf. Manage.*, vol. 4, no. 4, pp. 233–237, 2006.
[65] I. Cantador, P. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011)," in *Proc. ACM Conf. Recommender Syst. (RecSys)*, 2011, pp. 387–388.
[66] D. Benz, A. Hotho, and R. Jäschke, B. Krause, and G. Stumme, "Query logs as folksonomies," *Datenbank-Spektrum*, vol. 10, no. 1, pp. 15–24, 2010.
[67] K. Bischoff, C. S. Firan, W. Nejdl, and R. Paiu, "Can all tags be used for search?" in *Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2008, pp. 193–202.
[68] B. Krause, A. Hotho, and G. Stumme, "A comparison of social bookmarking with traditional search," in *Proc. Eur. Conf. Inf. Retr. (ECIR)*, 2008, pp. 101–113.
[69] E. M. Voorhees, "The TREC-8 question answering track report," NIST, Gaithersburg, MD, USA, Tech. Rep., 1999.

ZHENGHUA XU received the B.Eng. degree in information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009, the M.Phil. degree in computer science from The University of Melbourne, Australia, in 2012, and the Ph.D. degree from the Department of Computer Science, University of Oxford, in 2018.

He is currently a Research Associate with the Department of Computer Science, University of Oxford. His research interests include big data, machine learning, deep learning, data mining, social Web search, and recommender systems.

OANA TIFREA-MARCIUSKA received the B.Sc. degree in computer science from Alexandru Ioan Cuza University, Romania, in 2008, and the M.Sc. degree (European Master in computational logic) from the Free University of Bozen-Bolzano, Italy, and the Vienna University of Technology, Austria, in 2010, and the Ph.D. degree from the Department of Computer Science, University of Oxford, in 2017.

From 2016 to 2018, she was with the Alan Turing Institute and the University of Oxford as a Post-Doctoral Researcher and a Visiting Researcher, respectively. She is currently a Research Scientist with Bloomberg, London.

Dr. Tifrea-Marciuska was a recipient of awards and honors, including the EPSRC Doctoral Prize, the Google Europe Fellowship in Social Search (Google European Doctoral Fellowship), the Best Master Thesis Award, and the Google Anita Scholarship.

THOMAS LUKASIEWICZ received the Ph.D. degree in computer science from the University of Augsburg, Germany, in 1996, and the Dozent degree (venia docendi) in practical and theoretical computer science from TU Vienna, Austria, in 2001.

He is currently a Professor of Computer Science with the Department of Computer Science, University of Oxford, U.K., and a Turing Fellow with the Alan Turing Institute, London, U.K. His research interests are in artificial intelligence and information systems, including especially knowledge representation and reasoning, uncertainty in AI, machine learning, the Semantic Web, and databases.

MARIA VANINA MARTINEZ received the Ph.D. degree from the University of Maryland, College Park, under the supervision of V. S. Subrahmanian. She held post-doctoral position at the University of Oxford. She was a Research Assistant with the Information Systems Group, University of Oxford.

She is currently a full-time Researcher with the Institute for Computer Science and Engineering, CONICET, Universidad Nacional del Sur, and a Visiting Professor with the Universidad de Buenos Aires, Argentina. Her research interests include reasoning under uncertainty, inconsistency management in databases and knowledgebases, defeasible reasoning, and argumentation.

GERARDO I. SIMARI received the Ph.D. degree in computer science from the University of Maryland, College Park, in 2010. He was a Post-Doctoral Researcher with the Department of Computer Science, University of Oxford, U.K., in 2011, and later continued there as a Senior Researcher and a Fulford Junior Research Fellow with the Somerville College.

He is currently a Professor with the Universidad Nacional del Sur, Bahía Blanca, and a full-time Researcher with CONICET, Argentina. His research focuses on topics within artificial intelligence and databases, especially reasoning under uncertainty, ontologies, and preferences.

CHENG CHEN received the B.Sc. degree in computer science from Huazhong Agricultural University, China, and the Ph.D. degree from the Department of Computer Science, Beijing University of Posts and Telecommunications, China.

She is currently with the China Academy of Electronics and Information Technology, China. Her research interests include machine learning, data mining, mobile recommendation, and deep learning.