Personalized Recommender Systems on the Social Web

Zhenghua Xu

Department of Computer Science

Balliol College

University of Oxford

A thesis submitted in fulfilment of the requirements

of the Degree of Doctor of Philosophy

August, 2017

Abstract

An exponential increase in the usage of the World Wide Web (Web 2.0) has led to a wide collection of social platforms that harness people's online activities such as online shopping (e.g., eBay and Amazon), entertainment (e.g., YouTube and Last.fm), and social networking (e.g., Facebook and Twitter). This has led to an overabundance of information online, which is far too consuming for individual users to browse. As such, personalized recommender systems have emerged, which aim to tailor the online information presented to the user based on their preferences. Such recommender systems have been widely studied in the research community and are also commonly deployed in industry, e.g., music recommendation on Last.fm, item recommendation on Amazon.com, and friend recommendation on Twitter.

Despite the huge prevalence of using social annotations in personalized recommendation, a major outstanding problem is that in general they are redundant, sparse and ambiguous. Therefore, in this thesis, we first study and address this problem by proposing a recommendation-oriented deep neural model, in the context of tag-aware personalized recommendation. Here, we also propose hybrid deep learning and neg-

ative sampling to enhance the deep models' training efficiency. Furthermore, since users' real-time news preferences are sometimes correlated with their geographical context, we then investigate location-aware personalized news recommendation using geographical topic models, where topic feature modelling and the user's long-term personal interests are used to enhance the recommendation performance. Finally, to alleviate the need for powerful computing facilities to train deep recommendation models, we propose a novel ontology-based similarity to circumvent tag ambiguity and redundancy for applications in light-weight recommender systems on the Social Web.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Thomas Lukasiewicz. The work presented in this thesis would not have been possible without his support and assistance. I am deeply influenced and encouraged by his guidance on how to do the high quality and high impact research work. He has provided insightful suggestions and guidance for me on the research work in this thesis. He is the person who patiently taught me the research skills and led me to the right research directions.

Furthermore, I would like to thank Dr. Cheng Chen, Mr. Yishu Miao, and Prof. Xiangwu Meng for their suggestions, comments and contribution to the work on the deep tag-aware personalized recommendation. I would also like to further thank Dr. Cheng Chen and Prof. Xiangwu Meng for their help on the location-aware personalized news recommendation. Finally, I would like to thank Dr. Oana Tifrea-Marciuska, Dr. Maria Vanina Martinez, Dr. Gerardo I. Simari, and Dr. Cheng Chen for their suggestions and comments on the ontology-based personalized recommender systems.

I wish to give my thanks to my wife, Mrs. Shan Sun, and my parents, Mr. Jinlian Xu and Mrs. Meilian Chen, for their continuous supports, physically, psychologically

and financially. They gave me a lot of encouragements and suggestions whenever I had difficulties in my research and my daily life. It is their help and love that enabled me to conduct my research work with confidence and determination. I also want to use this thesis as the 100-day gift of my newborn baby son, Mr. Chengfan Xu, who brings me the strength and faith to work hard to be a good role model of his life.

I would like to thank my college, Balliol College, who not only offered me the Jason Hu Scholarship to support my DPhil research, but also tried its best to help me whenever I needed it. I would also like to thank the Department of Computer Science and University of Oxford for giving me such an excellent research platform; I believe all the things I learned here will be the greatest wealth of my life.

CONTENTS

CHAPTER 1

Introduction

In recent years, there has been an exponential increase in the popularity of the Social Web (Web 2.0), 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: online shopping (e.g., on eBay and Amazon), entertainment (e.g., on YouTube and Last.fm), and social networking (e.g., on 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. 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 [31, 35, 54, 85]; and commercial applications have been widely deployed, e.g., music recommendation on Last.fm, item recommendation on Amazon.com, and friend recommendation on Twitter.

To realize personalized recommendations, recommender systems are typically built based on using 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 [147]. 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 feedbacks for the given recommendation results [111]. The quality of the resulting user profile is 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 query history [87, 146, 149, 154], browsing history [157], the user's current tasks [109] or intents [159], and even eye-tracking during search sessions [78] 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 benefit reasons, these data are usually not accessible for 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. For their own benefits, 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 [178, 195]:

- 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.

Given this status quo, in recent years, there has been a growing number of research efforts exploiting the use of social annotations in personalized recommendations, based on content-based filtering [32, 33, 164, 178], collaborative-filtering [160, 197], or graph-based ranking techniques [70]. Nevertheless, relevant research in this area is still only just beginning. The goal of this thesis is to develop effective and efficient personalized recommender systems on the Social Web using social annotations, with focus on addressing the outstanding research problems in the following three important sub-fields:

- tag-aware personalized recommender systems,
- location-aware personalized news recommender systems,

- ontology-based personalized recommender systems.

1.1 Tag-Aware Personalized Recommender Systems

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. This practice is called *social tagging* or *collaborative tagging*, while the classification systems derived from social tagging activities are known as *folksonomies*. 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* [30, 33, 148, 164, 178, 203] or *collaborative filtering* [32, 160]. To this end, content-based filtering is especially important and attracts more and more attention in the research community [30, 33, 164, 178, 203], because it allows recommender systems to incorporate tags as additional content information to improve recommendations [26].

To achieve personalization in content-based tag-aware personalized recommendation, a similarity measure is required to estimate the relevance of users' preferences (described by user profiles) and the social summaries of items (described by item profiles). A vector space model (VSM) [138] is usually used to represent both as weighted vectors of tags. However, since users can freely choose their own vocabulary of tags, the resulting social tags may contain many uncontrolled vocabularies, such as homonyms (tags with the same spelling but of different meanings), synonyms (different tags of the same meaning), words in arbitrary languages, or even user-created words. This results in very sparse, redundant, and ambiguous tag information and leads to inaccurate similarity values between corresponding profiles. Consequently, the perfor-

mance of content-based tag-aware recommendation systems is greatly degraded. This is also called as the *uncontrolled vocabulary problem* on the Social Web.

A solution to this problem is to apply clustering in the tag space [148]; however, clustering necessarily requires computing the similarity between tags, which is usually very time-consuming. Another solution is to use autoencoders. In [203], abstract feature representations for tag-based user profiles are first modeled by autoencoders and then used as inputs to user-based collaborative filtering to generate recommendations. Although this method was reported to achieve better performance than the clustering-based collaborative filtering method [203], it still suffers from the following drawback: the model's learning signal is based on reconstruction errors, which are not directly related to the ability to distinguish the user's target items from irrelevant ones (the ultimate objective of personalized recommendation). As such, although the resulting abstract feature representations are quite effective for reconstruction, they are not very effective for personalized recommendation. Consequently, its improvement in personalized recommendation is not very significant.

Motivated by the above observations, we propose to address the uncontrolled vocabulary problem by using deep neural networks to map the tag-based user and item profiles to an abstract deep feature space, where the similarities between users and their target items (resp., irrelevant items) are maximized (resp., minimized). We call the similarities in the deep feature space *deep-semantic similarities* and this model *deep-semantic similarity-based personalized recommendation* (DSPR) model.

In general, DSPR has the following advantages:

- The deep model is trained using a recommendation-oriented learning signal, which is directly correlated with differentiating the user's target items from the irrelevant ones. The resulting abstract features for user and item profiles are

therefore very effective representations for personalized recommendation.

- Deep neural networks in DSPR extract more abstract and denser features layer-by-layer, so DSPR overcomes the sparsity and redundancy problems in the social tag space.

- Also, the input synonyms (resp., homonyms) have similar (resp., different) impact on output features of DSPR, which addresses the ambiguity problem in the social tag space.

Interpretability (or transparency) is a common concern for some deep learning applications, but this problem is remedied in DSPR. Specifically, if we define the most influential tags of an output abstract feature as input tags whose weight changes will most influence the weight of this output feature, the semantics of output features can be inferred by using their corresponding most influential tags. For example, if the tags "www3", "Web", and "online" are the top 3 most influential tags of an output abstract feature, it is reasonable to infer that this feature is about "Internet"; consequently, by seeing a user's output abstract profile with a high weight for this feature, we can interpret that this user has strong interest on "Internet".

Despite achieving superior performance, the training of the DSPR model is usually very time-consuming in practice, mainly because DSPR has many hidden layers and a huge number of candidate items. On the one hand, the many hidden layers in DSPR lead to minuscule and insignificant learning signals for the first few layers, which require numerous different training runs to reach model convergence. This problem is also known as *diffusion or vanishing gradients*. On the other hand, to train DSPR, the deep-semantic similarities between the user in each training sample and all the candidate items have to be computed in each training run. Since the number of candidate

items for an online recommendation system is usually very large (millions) and training deep neural networks often requires many training samples, the processing of each training run is computationally very expensive, resulting into a *scale problem*.

Therefore, we further propose to solve these problems by *hybrid deep learning* and *negative sampling*. With both improvements, the resulting hybrid deep learning-based personalized recommendation with negative sampling (HDLPR-NS) model becomes very efficient in model training (hundreds of times quicker than DSPR), whilst maintaining superior recommendation performance.

1.2 Location-Aware Personalized News Recommender Systems

We also note that although users' long-term preferences mainly rely on their personal interests that are usually learnt from their historical activities, e.g., social tags, their real-time preferences on some specific kinds of objects can evolve with the change of locations. These special kinds of objects are called *location-sensitive objects*.

Generally, the relevance between locations and most location-sensitive objects, e.g., events, stores, or facilities, lies on the physical distance between the user and the candidate objects, which usually have high influence on the user's real-time choice. For example, a user may choose a street parking spot nearby instead of driving 10 km to park at a secured parking lot, though he/she generally prefer the latter. Furthermore, it is often easy to obtain GPS information for these objects. Therefore, most of the classic location-aware recommendation solutions are physical distance-based [40, 53, 130, 190, 200], where the locational relevance of an object to a user is measured by the physical distance between their locations, in the form of GPS coordinates.

News articles are also location-sensitive objectives: users' real-time news preferences are sometimes correlated with their geographical context. For example, A stock trader's long-term news preferences are financial and sports news; however, he may focus on financial news when working at the Wall Street and leave sports news to be read at home; also, he may like to receive news recommendation about the change of the local weather when strolling at park, though he usually does not care about this.

However, the above two properties are not held for news articles. On the one hand, the descriptions of event locations in many news articles are vague and general (mentioning only a city or suburb), making it very difficult or even impossible to obtain accurate GPS information for this kind of news. On the other hand, even with GPS coordinates, it is not sound to assume that a news happening nearby will be more attractive than another news happening 10 km away. Actually, the relevance of news to locations actually lies in the relevance of the topics of news to the geographical topics of locations. For example, an Apple store can have the geographical topics "iPhone" and "shopping", which enhance the priority of recommending news about big sales of Samsung mobiles to people in the store.

Consequently, the widely studied physical distance-based location-aware recommendation solutions are not adequate to be applied in news recommendation. Indeed, although there are some attempts at this direction [11, 12, 23, 131, 135], few of their results is convincing. This is thus degrading the research popularity of location-aware news recommendation: there are a lot of recommender systems that use location information, but few of them are focusing on the applications in news articles.

A recent effort to cope with the special topic-based location relevance of news is to extract the geographical topics of a location; then the relevance of a news to a location is estimated according to the similarity of their topics. This is called *geographical-*

topic-based location-aware news recommendation. Obviously, the quality of topic representations of locations in this recommendation task is crucial. A range of topic models are thus proposed to learn effective geographical topics from locations, such as latent Dirichlet allocation (LDA) [24], explicit semantic analysis (ESA) [51], probabilistic latent semantic analysis (pLSA) [202], and their improved models [116, 152, 153, 185, 186], which are called *geographical topic models*. Among all the geographical topic models, *explicit localized semantic analysis (ELSA)* [152] was reported to achieve the best performance in geographical-topic-based location-aware news recommendation.

However, we also note that, as the state-of-the-art for this recommendation task, ELSA only attracts a few dozens citations and has no practical applications. We believe this is mainly because of the following two drawbacks of ELSA.

- In ELSA, news and locations are projected to a Wikipedia-based topic space, where each Wikipedia concept is regarded as a possible topic. Due to the huge volume (millions) of concepts on Wikipedia, the Wikipedia-based topic space in ELSA is very high dimensional, sparse, and redundant. The high dimensionality problem thus reduces the efficiency of ELSA-based recommendation models, while the sparsity and redundancy degrades the recommendation accuracy.

- ELSA takes into account only the users' geographical information for news recommendation and ignores their personal news interests. This is, however, unreasonable for real-world situations. In fact, users' real-time news preferences are influenced by both their long-term personal interests and their real-time geographical context. For example, a housewife may prefer to read entertainment news at home, while the husband may like sports news at the same place. This

difference comes from their long-term personal interests, which thus should also be considered in location-aware news recommendation.

Therefore, motivated by these observations, we propose to overcome the drawbacks of ELSA and to further improve the performance of geographicl-topic-based location-aware news recommender systems by the following solutions: We first propose to apply machine learning techniques to resolve high-dimensionality, sparsity, and redundancy problems in the Wikipedia-based topic space in ELSA. We then extend ELSA to also take into account the user's long-term personal interests to achieve better personalized news recommendation. Finally, a hybrid location-aware personalized news recommendation model that combines and has the advantages of the above two improvements is proposed.

1.3 Ontology-Based Personalized Recommender Systems

Although the above deep-learning-based solutions, e.g., HDLPR-NS and LP-DSA, can solve the ambiguity and redundancy problems and achieve superior recommendation performances on the Social Web, they both share the following shortcoming. In order to achieve good recommendation performances, deep-learning-based recommendation solutions must always be trained well using a large amount of data, and this training process needs to be invoked frequently to capture the dynamic changes on the Social Web. Very powerful computing facilities are therefore required to support online recommender systems, which would not be generally available for small-size startup companies or independent developers, though they are not a problem for tech giants, e.g., Facebook or Amazon.

To address this problem, in the last part of my thesis, we leverage ontology tech-

niques [58] and propose ontology-based personalized recommender systems, which utilize a novel similarity metric, called *ontological similarity (OntSim)*, to tackle tag ambiguity and redundancy problems without the need of model training. The novelty of this ontological similarity is to use concept similarities in ontologies to estimate the semantic similarities of tags in profiles, which are semantically more accurate than those generated by 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, it is a complement to deep-learning-based recommendation solutions, and will be a good disambiguation choice for light-weight recommender systems.

1.4 Contributions

Generally, this thesis studies and addresses the outstanding research problems in the context of personalized recommender systems on the Social Web using social annotations. The detailed contributions are as follows.

First, we propose to use the deep learning techniques to solve the uncontrolled vocabulary problem on tag-aware personalized recommender systems, making the following contributions:

- We propose a deep-semantic similarity-based personalized recommendation (DSPR) model; DSPR utilizes deep neural networks with a recommendation-oriented learning signal to solve the uncontrolled vocabulary problem in social tags, and greatly enhances the recommendation accuracy of tag-aware personalized recommender systems.

- To ensure scalability in practice, we further propose a hybrid deep learning-based

personalized recommendation with negative sampling (HDLPR-NS) model. HDLPR-NS solves the vanishing gradients problem using hybrid deep learning, which directly integrates autoencoders with the neural networks of DSPR to generate additional learning signals based on reconstruction errors. For the scale problem, to reduce the processing time needed for each training run, HDLPR-NS adopts *negative sampling* [112] to randomly sample a small number of negative examples to approximate the noise.

- We conduct an extensive experimental study using two real-world social tagging datasets, Delicious and Last.Fm. The experimental results show that (i) DSPR and HDLPR-NS both significantly outperform the state-of-the-art baselines in tag-aware personalized recommendation (3.8 times better than the best baseline), and (ii) by using hybrid deep learning and negative sampling, the model training of HDLPR-NS is hundreds of times more efficient than DSPR, while achieving a similar (and sometimes even better) training quality and recommendation performance.

Second, we further investigate location-aware personalized news recommender systems based on geographical topic models.

- We first propose three novel geographical topic feature models, *clustering-based localized semantic analysis (CLSA)*, *autoencoder-based localized semantic analysis (ALSA)*, and *deep localized semantic analysis (DLSA)* to solve high-dimensionality, sparsity, and redundancy problems in ELSA by *topic feature modelling*. These three models integrate clustering, autoencoder, and recommendation-oriented deep neural networks, respectively, with ELSA to obtain dense, abstract, low dimensional, and effective topic feature representations for locations and news,

and thus achieve more effective and efficient location-aware news recommendations than ELSA.

- We also propose to take into account the users' long-term personal interests to extend ELSA to a *location-aware personalised news recommendation with explicit semantic analysis (LP-ESA)* model. LP-ESA offers news recommendation based not only on the user's current location, but also on his/her long-term personal interests, where a collection of user history data (e.g., tweeting history) is used to model the given user's interests, and collections of geo-tagged documents are used as descriptions of locations.

- As LP-ESA is also an ESA-based method, similar to ELSA, it also suffers from high-dimensionality, sparsity, and redundancy problems. Therefore, we finally propose a hybrid model which integrates the intuitions of DLSA and LP-ESA to also exploit recommendation-oriented deep neural networks to overcome these problems in LP-ESA. This hybrid model is called *location-aware personalized news recommendation with deep semantic analysis (LP-DSA)* model, which is a deep learning extension of LP-ESA as well as a personalized extension of DLSA.

- Extensive experiments are conducted using a public real-world dataset. The results show that: (i) The proposed CLSA, ALSA, and DLSA models all greatly outperform the state-of-the-art geographic topic model, ELSA, in location-aware news recommendations in terms of both the recommendation effectiveness and efficiency, where DLSA further outperforms CLSA and ALSA in recommendation effectiveness. (ii) LP-ESA significantly outperforms the baselines, ELSA and STPM, in terms of all evaluation metrics. (iii) The hybrid model LP-DSA, that integrates DLSA and LP-ESA, is capable to take the advantages of both

models and achieve better recommendation effectiveness than DLSA and LP-ESA. (iv) With the help of deep neural networks, LP-DSA and DLSA have much lower online recommendation time-cost (about 25 times quicker) than LP-ESA and ELSA.

Third, as a complement to deep-learning-based recommendation solutions, we finally propose ontology-based personalized recommender systems which creatively utilize ontological similarity to tackle tag ambiguity (e.g., homonyms) and redundancy (e.g., synonyms) problems without the need of model training. The detailed contributions are as follows:

- The novelty of the ontological similarity is to use concept similarities in ontologies to estimate the semantic similarities of tags in profiles, which are semantically more accurate than those generated by 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.

- A two-step top-down disambiguation algorithm is first developed to solve the tag ambiguity and redundancy problems 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 the ontological similarity for content-based tag-aware personalized recommendation is proposed. The computational complexity of these algorithms is also given and proved.

- We conduct extensive experiments based on a public dataset and evaluate the performance of the proposed algorithms from three perspectives: tag disambiguation, tag-to-ontology allocation, and personalized recommendation. The

experimental results show that (i) the proposed ontology-based solution greatly outperforms (more than double performance) the state-of-the art baselines in tag disambiguation; (ii) 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; (iii) 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; and (iv) although HDLPR-NS generally achieves higher accuracy in tag-aware recommendation, it also has much higher computational cost; therefore, ontological similarity is a reasonable disambiguation alternative for light-weight recommender systems without powerful computing capability.

Finally, we compare the proposed recommendation models to analyze their pros and cons and also give some advice on their potential extensions and applications.

1.5 List of Publications

Six papers that are published or submitted during the period of my DPhil study are summarized as below. The work on deep tag-aware personalized recommender systems (presented in Chapter 4) has been published in IJCAI 2017 [3] and CIKM 2016 [5]. The work on location-aware personalized news recommender systems (presented in Chapter 5) has been published in IEEE Aceess [2] and DASFAA 2017 [4]. The work on ontology-based personalized recommender systems (presented in Chapter 6) has been finished and submitted for publication in ACM TOIT [1].

[1] **Zhenghua Xu**, Oana Tifrea-Marciuska, Thomas Lukasiewicz, Maria Vanina Martinez, Gerardo I. Simari, and Cheng Chen. Personalized Search and Recommendation Based on Ontological Similarity in Folksonomies. Submitted to *ACM Transactions on Internet Technology (TOIT)*.

[2] Cheng Chen, Xiangwu Meng, **Zhenghua Xu** and Thomas Lukasiewicz. Location-Aware Personalized News Recommendation using Deep Semantic Analysis. *IEEE Access*, 2017.

[3] **Zhenghua Xu**, Thomas Lukasiewicz, Cheng Chen, Yishu Miao and Xiangwu Meng. Tag-Aware Personalized Recommendation Using a Hybrid Deep Model. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[4] Cheng Chen, Thomas Lukasiewicz, Xiangwu Meng and **Zhenghua Xu** (in alphabetical order). Location-Aware News Recommendation Using Deep Localized Semantic Analysis. In *Proceedings of the 22nd International Conference on Database Systems for Advanced Applications (DASFAA)*, 2017.

[5] **Zhenghua Xu**, Cheng Chen, Thomas Lukasiewicz, Yishu Miao and Xiangwu Meng. Tag-Aware Personalized Recommendation Using a Deep-Semantic Similarity Model with Negative Sampling. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management (CIKM)*, 2016.

[6] **Zhenghua Xu**, Thomas Lukasiewicz and Oana Tifrea-Marciuska. Improving Personalized Search on the Social Web Based on Similarities Between Users. In *Proceedings of the 8th International Conference on Scalable Uncertainty Management (SUM)*, 2014.

1.6 Outline

The rest of this thesis is organized as follows:

- Chapter 2 reviews the related literature and then compares and clarifies the differences between the state-of-the-art solutions and the proposed methodologies in all three sub-areas.

- Chapter 3 presents preliminaries, including the following critical concepts and technologies: folksonomy, vector space model, personalized recommendation, similarity measures, clustering, and autoencoders.

- Chapter 4 proposes a deep tag-aware personalized recommendation model, called deep-semantic similarity-based personalized recommendation (DSPR) model, to solve the uncontrolled vocabulary problem in social tags. Moreover, to ensure scalability in practice, we further propose to improve DSPR by using hybrid deep learning and negative sampling to achieve a very efficient deep model training; the resulting model is called hybrid deep learning-based personalized recommendation with negative sampling (HDLPR-NS) model. Finally, the experimental study of DSPR and HDLPR-NS is also shown in this chapter.

- Chapter 5 focuses on location-aware news recommendation. We first identify two drawbacks of the state-of-the-art location-aware news recommendation model, ELSA. Then we propose three novel geographical topic feature models, CLSA, ALSA, and DLSA, and a location-aware personalized news recommendation with explicit semantic analysis (LP-ESA) model, to overcome these two shortcomings. Finally, we propose a hybrid model, LP-DSA, which is an integration of ideas of LP-ESA and DLSA, and has the advantages of both improved models.

The experimental studies on the proposed location-aware news recommendation models are also shown in this chapter.

- Chapter 6 proposes ontology-based personalized recommender systems, which utilize a ontological similarity measure to tackle the tag ambiguity and redundancy problems without the need of model training. In this chapter, we first develop a two-step top-down disambiguation algorithm to solve the tag ambiguity and redundancy problems 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 the ontological similarity for content-based tag-aware personalized recommendation is proposed. The computational complexity of these algorithms is also given and proved. We also perform extensive experiments based on a public dataset and evaluate the performance of the proposed algorithms from three perspectives: tag disambiguation, tag-to-ontology allocation, and personalized recommendation, in this chapter.

- Chapter 7 compares the proposed recommendation models and gives advice on their relative merits and potential applications.

- Chapter 8 concludes the work in this thesis and discusses future work.

CHAPTER 2

Literature Review

2.1 Recommender Systems

The past decades have witnessed a rapid growth of the World Wide Web, resulting in an explosion of online information. Consequently, it is not possible for users to go through all the information online to find out the really relevant ones. Therefore, recommender systems, which filter out irrelevant online information and recommend to users their preferred online resources, have been widely deployed in the industry, e.g., music recommendation on Last.fm, item recommendation on Amazon.com, and friend recommendation on Twitter, and also attracted many research efforts in the areas of music recommendation [42, 88, 115, 141, 158], movie recommendation [36, 95, 150, 170, 196], TV recommendation [16, 94, 129, 189], news recommendation [69, 80, 107, 187], book recommendation [46, 117, 151], document recommendation [126, 127, 134, 144], e-commence recommendation [37, 74, 100], e-learning recommendation [28, 47,

60], and medical recommendation [68, 75, 193].

According to the different technologies used for recommendation, recommender systems can be classified into two types: content-based filtering and collaborative filtering. Content-based filtering systems [86, 128, 137, 183] utilizes the content information of items to recommend the items that are similar to those a user liked previously, where the similarities between user and item profiles are computed to evaluate their relevance. Collaborative filtering systems recommend to users items liked by similar users using K nearest neighbor (KNN) [25, 63, 82] or model-based solutions, e.g., matrix factorization [81, 108, 193], neural networks [45, 73, 173], and Bayesian classifier [106, 121, 191].

As the first step of personalized recommendation, recommender systems have to collect information about users' preferences or interests, which are then used to personalize the recommendation list, i.e., recommended items are re-ranked to put items that are more relevant to the user's personal interests higher in the recommendation list [147]. Users' preferences or interests can be acquired by leveraging their information that is either explicitly or implicitly provided.

Explicit user preferences are usually offered by the users directly by filling in preference information or providing positive or negative feedbacks for the given recommendation results [111]. For example, users are requested to label items as "relevant" or "not relevant" in [20], while numerical and symbolic ratings are given by users in [145] and [122]. Consequently, the quality of the resulting user profile is strongly correlated with the user's willingness and capability to provide sufficient and appropriate preference information, which thus results in very unstable recommendation performances.

Consequently, as an effortless alternative, implicit user activity data, such as query

history [87, 146, 149, 154], browsing history [157], the user's current tasks [109] or intents [159], and even eye-tracking during the search session [78] have been widely used in the recommender systems of modern websites and applications, such as Google, Amazon, Youtube and Facebook, to provide more information about user preference. However, due to privacy, ethical, legal, or benefit reasons, these data are usually not accessible for external parties. Specifically, by accepting a "user agreement", users may allow the service provider to collect and use their implicit activity data to create better user experience, but it is normally prohibited to share these data with third parties. Furthermore, for their own benefits, the IT companies are also generally reluctant to release these data to other parties. Therefore, the need of publicly available and easy accessible user data is compelling for the development of personalized recommender systems on the Social Web.

Social annotations are ideal data for privacy-enhanced personalization for the following reasons [178, 195]: (i) Social annotations are usually publicly available online, so they are generally easily accessible (with the permission of users and/or website owners). (ii) Social annotations are provided by users directly as their individual opinion about an online item, so the interests and preferences of users can be harvested by the aggregation of their social annotations. (iii) 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. Consequently, as a privacy-enhanced personalization technique, more and more research efforts have been put in personalized recommendation using social annotations [31, 35, 54, 85].

2.2 Recommendation Based on Deep Learning

Deep learning, also called deep structure learning, refers to a family of machine learning techniques that have a deep architecture. In the past decades, deep learning models have repeatedly outperformed traditional shallow models, such as support vector machine (SVM) [118] or logistic regression [123], in many application domains, such as computer vision and speech recognition [198].

To our knowledge, the first work that introduce deep learning to recommender systems is [136], where a restricted Boltzmann machine is used for collaborative-filtering-based recommendation. Since then, a race has been witnessed in both academia and industry to apply deep learning in various recommendation tasks [192], e.g., point-of-interest (POI) recommendation [162, 163, 167, 181], music recommendation [77, 165, 169], image recommendation [90, 176], and vedio recommendation [45, 136]. In addition, an annual deep learning for recommender systems (DLRS) Workshop has been organized since 2016 [79].

In the rest of this section, I will review and summarize the existing deep-learning-based recommendation models, which are grouped according to the type of deep learning techniques. Furthermore, comparison will be done to the closely related works to clarify their differences to the deep recommendation models proposed in this thesis.

2.2.1 Multilayer Perceptron

A multilayer perceptron (MLP) is a feed-forward neural network, consisting of an input layer, an output layer, and multiple hidden layers. Due to its simplicity and efficiency, MLP is widely studied in the research community [59, 62, 103, 168] and also deployed for commercial uses [41, 45].

In [62], a neural collaborative filtering model is constructed by combining MLP with collaborative filtering to capture the relationship between users and items. This model is then extended in [103] and [168] for cross-domain recommendations. [41] also proposes a wide and deep learning model to employ MLP in recommending Apps in Google play; the model has a wide learning part (consisting of a single layer perceptron), aiming to capture direct features, and a deep learning part (consisting of a multilayer perceptron), aiming to obtain abstract features. Chen et al. [39] extend [41] with local connectivity and significantly enhance the model's running efficiency. A deep factorization machine (DeepFM) is proposed in [59] with a similar motivation to [39] and [41]; DeepFM uses a MLP and a factorization machine to learn high-order and lower-order features, respectively. Besides App recommendation in [41], MLP is also used in [45] for video recommending in YouTube.

2.2.2 Convolutional Neural Network

Convolutional neural networks (CNNs) are a type of neural network that has convolution and pooling layers. CNN is capable to efficiently extract both global and local features from textual, image, and audio data.

CNN is used in [57] to recommend hashtags from microblogs. Besides a global channel consisting of convolution and max-pooling layers, the model has a local attention channel, where words that are less informative is filtered out before going to the next layer. Later, this model is also used in [143] (with some modifications on the last few layers) to extract features from review texts. [199] proposes to combine CNN with a factorization machine, which is applied as the final layer to capture the interactions between users and items. Similarly, CNN is integrated with probabilistic matrix factorization in [81] to learn more accurate feature representation with the help of contextual

information.

Besides textual data, CNN is also used to extract image features for POI recommendation in [167] and audio features for music recommendation in [43] and [165]. Although the recommendation in [43] is also hybrid, it is very different to the proposed hybrid model HDLPR-NS: the things combined in [43] are the recommendation strategies (it uses both collaborative and content-based filtering), while HDLPR-NS is claimed to be hybrid, because it integrates two deep learning models (autoencoders and deep semantic similarity models).

2.2.3 Recurrent Neural Network

Recurrent neural networks (RNNs) are a special kind of neural network that has loops to remember previous information, which is good at capturing temporal or sequential features. Long short term memory (LSTM) and gated recurrent unit (GRU) are the most commonly used variants to avoid vanishing gradients.

Wu et al. [172] propose a recurrent recommender network to learn the seasonal feature evaluation of items and users over time. This model is further improved in [171] to learn the features of ratings and reviews using LSTM. Bansal et al. [10] propose to apply GRU for text recommendation, where text sequences are encoded into latent factor model by GRU and a multi-task learning strategy is used to overcome the sparsity problem and to avoid over-fitting. Similarly, GRU is also used for news recommendation in [119], where it generates user representations from browsing histories.

Almahairi et al. [8] propose to use recurrent neural networks to model the representations of user reviews, which are then used as a regularizer of a matrix factorization model for recommendation. It is interesting that, differently from most other deep-learning-based recommender systems, the use of RNN in [8] does not lead to

an improved performance. I believe this is because RNN is used for a very different purpose, i.e., regularization, in [8]. As we know, a strong point of deep learning models is their excellent capability in modelling effective feature representations, which is usually very helpful to improve the recommendation performance. However, a very effective and representative feature model is not necessary to be a good regularization model. In contrast, great modelling power may undermine the model's ability to act as a regularizer; this is because regularization aims to bring in some noise and flexibility, but the feature representations from deep models may be too accurate and certain to cope with that.

2.2.4 Autoencoder

Autoencoders are unsupervised neural models that reconstructs its input data in the output layer and learns abstract feature representations of the input data in the code layer. Therefore, autoencoders are utilized in recommender systems to either predict unknown ratings or learn abstract feature representations.

Sedhain et al. [142] propose an autoencoder-based recommender system, where a partially filled user-item rating matrix is sent to autoencoders as inputs. By minimizing the reconstruction errors of the known ratings, this model is capable to predict the unknown ratings using the reconstructed rating matrix in the output layer. [142] is extended by [155] and [156], which propose a hybrid recommendation model that integrates collaborative filtering with autoencoders; this model overcomes the sparsity and cold start problems using user profiles and item descriptions as additional information, and it is also more robust by denoising. Besides rating prediction, the denoising autoencoder is also used for ranking prediction in [174]. These works all have a different research purpose to our HDLPR-NS model: HDLPR-NS aims to efficiently learn ab-

stract feature representations in the code layers of autoencoders, instead of predicting ratings/ranking.

In [101], a deep collaborative filtering framework is proposed to improve the performance of personalized recommendation by using autoencoders for feature representation learning. This model differs from HDLPR-NS in the following perspectives: (i) [101] is not a tag-aware recommendation model, (ii) [101] is based on collaborative filtering and our HDLPR-NS is content-based, and (iii) the model in [101] is trained to minimize both the reconstruction errors in autoencoders and the matrix factorization errors, such that two components both significantly influence the model's training quality; however, as shown in our experimental study in Section 4.3, the recommendation-oriented learning signal has the dominant effect on the training quality of the HDLPR-NS model, while the reconstruction-error-based signal is mainly for enhancing training efficiency.

Recently, [203] proposes another recommender system that utilizes autoencoders to learn abstract feature representations of users. In [203], a user profile is first represented as vectors over tags; to overcome the sparsity, redundancy, and ambiguity problems, the user profiles are sent to an autoencoder to learn abstract latent representation of users; based on the resulting abstract user profiles, a user-based collaborative filtering is conducted for recommendation, i.e., the unknown rating of a user to an item is estimated according to the ratings of his/her top-k most similar users to this item. Among all the existing autoencoder-based recommender systems, [203] is the only one that specially focuses on tag-aware recommendation and the uncontrolled vocabulary problem in social tags space. Therefore, it is considered as the state-of-the-art in this research question and used as a baseline to evaluate the recommendation performance of our proposed DSPR and HDLPR-NS models.

2.2.5 Other Related Work on Deep Learning

Similarly to DSPR and HDLPR-NS, deep-semantic similarity models are also used in [72] for personalized recommendation using click-through data. Elkahky et al. [49] then extend this model to a multi-view deep learning model for cross-domain user modelling in recommendation. However, these models are very different from DSPR and HDLPR-NS: (i) they are not tag-aware systems and not designed to solve the redundancy, sparsity, and ambiguity problems in the tag space; (ii) hybrid deep learning is not applied in these works to accelerate the convergence of the model; (iii) instead of using negative sampling, these two works intentionally assume the number of candidate items to be very small (5 in [72] and 10 in [49]) to make the model trainable. Obviously, this assumption is unreasonable in real-world situations. Negative sampling is first introduced in the NLP community to learn word representation more efficiently [112]. To our knowledge, this is the first work which applies hybrid deep learning and negative sampling to enhance the training efficiency of deep-semantic similarity-based recommendation or search.

There exists a fundamental problem of using back-propagation to train neural networks with many hidden layers: when learning signals are backpropagated to the first few layers, they become minuscule and insignificant, resulting in a very slow learning progress and high probability of falling into bad local minima. A widely used solution is to pre-train each layer before using back-propagation to fine-tune the entire deep neural network [50], where the pre-training is usually done by restricted Boltzmann machines (RBMs) [64, 65] or autoencoders [18]. Instead of using autoencoders for pre-training, in Chapter 4, we propose to overcome the bottleneck of back-propagation by directly integrating autoencoders into the network to generate additional learning signals based on reconstruction errors. Here, hybrid deep learning is used individually

to accelerate the deep model's learning progress; but it can also be used as a complement of pre-training to speed up the learning process in the "fine-tuning" step.

2.3 Tag-Aware Personalized Recommendation

As for tag-aware personalized recommender systems, content-based filtering is especially important, as it allows recommender systems to incorporate additional tag information to help recommendation [26]. Consequently, content-based tag-aware personalized recommendation has been widely studied [30, 33, 164, 178, 203], where the vector space model (VSM) [138] is the most widely adopted data model to construct user and item profiles.

Specifically, Xu et al. [178] and Cantador et al. [33] propose to use the cosine similarity of tag-based user and document profiles to personalize search and recommendation results on the Social Web, respectively. The work by Xu et al. [178] is then extended by Bouadjenek et al. [30], 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). In addition, Vallet et al. [164] propose to use the 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.

However, due to uncontrolled vocabularies, social tags are usually redundant, sparse, and 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 [148], 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 another solution is to use autoencoders [203], due to their capability to extract abstract representations [17]. As they are the state-of-the-art solutions for the same problem as the proposed DSPR and HDLPR-NS, these two methods are used as baselines in the experiments in Chapter 4.

Besides content-based filtering solutions, tag-aware personalized recommendation can also be achieved using collaborative filtering [32, 160, 197], graph-based ranking approaches [56, 70], or hybrid approaches [55]. For example, [197] proposes to use model-based collaborative filtering based on probabilistic matrix factorization for tag-aware recommendation. In addition, a graph-based ranking solution, called FolkRank is proposed in [70], which extends the PageRank algorithm to folksonomies, where tags, users, and items are treated as nodes and are connected via assignments, and then a weight passing scheme is used to derive the importance of these nodes.

2.4 Location-Aware Personalized News Recommendation

With the rapid development of the mobile network, more and more people tend to access the Social Web via their mobiles or other handheld devices, e.g., tablets. The wide use of mobile phones with positioning technology makes it easy for people to broadcast their locations, which are very valuable information for location-aware recommendations [13, 91, 121, 140]: users' location information can be used to accurately recommend news articles [11, 180], events/activities [130, 200], friends [99, 139, 201], advertisements [96, 182], tours [53, 188], or facilities [40, 190] according to the geographical contexts of users.

Content-based news recommendation aims to recommend to users the news that match their news interests best [44, 97]. Users' interests in news are usually modelled by their explicit ratings or browsing history (e.g., visited pages, reading times, and downloading records) [21]. Both heuristic [7] and model-based methods [184, 89] are proposed for content-based news recommendation: the former are mainly based on mathematical or statistical measurements (e.g., cosine similarity and Euclidean distance), while the latter make use of machine learning techniques or mathematical models (e.g., Bayesian networks and decision trees). For example, Abel et al. [7] proposes to combine news with information on the social media (e.g., tweets) to construct three kinds of user profiles, and then computes the cosine similarity between user profiles and news articles for personalized news recommendation. Yeung et al. [184] uses Bayesian networks to predict levels of users' interesting news categories, which are then used to provide real-time personalized news recommendation; while a decision tree model is utilized to predict user preferences for news recommendation in [89].

However, in the era of mobile and wireless networks, users' news preferences are also influenced by their geographical contexts, i.e., people usually pay more attention to the news that happened nearby than those far from them. Therefore, more and more research efforts have been put into location-aware news recommendation, which mainly focuses on two research directions: physical distance-based [11, 12, 23, 131, 135] and geographical topic-based [24, 51, 152, 153, 186].

As for physical distance-based news recommendation, GeoFeed [11] and GeoRank [12] recommend to users some news articles happening at the users' current locations or within a given range, where GeoRank uses only static location points of both users and news, and GeoFeed allows news with spatial extent; Pedro et al. [135] utilizes the Euclidean distance between user and news locations to measure the impor-

tance of news; LocaNews [23, 131] provides three versions of news and offers to users the most suitable ones according to their different distances to the news locations; Wen et al. [179, 180] proposes a news stream recommendation framework, called MobiFeed, to further study news recommendation according to users' moving tracks.

However, in the real-world context, the description of event locations in many news articles are very vague and general (mentioning only a city or suburb); so, obtaining accurate GPS information for this kind of news is very difficult and sometimes even impossible. Consequently, the application of physical distance-based methods is heavily limited.

As for more generic location-aware news recommendation, geographical topic-based methods are proposed [24, 51, 152, 153, 186]. Instead of using GPS coordinates, the locations are described using topic vectors, and the relevance of a news article to a user is measured by the similarity between the topic vectors of the news and the current location of the user. Therefore, the topic representations of locations are crucial for topic-based location-aware news recommendation, and a range of topic models, such as latent Dirichlet allocation (LDA) [24], explicit semantic analysis (ESA) [51], probabilistic latent semantic analysis (pLSA) [202], and their improved models [116, 152, 153, 185, 186], has been used for geographical topic-based location-aware new recommendation. For example, Son et al. [153] uses probabilistic explicit semantic analysis (PESA), an improved ESA geographical topic model, to extract geo-topics based on Wikipedia concepts and the links between concepts. Zhou et al. [202] uses a multi-modal location-dependent pLSA (mLD-pLSA) to learn geo-topics from location-related news and visible content (e.g., images).

The state-of-the-art geographical topic-based location-aware news recommendation method is explicit localized semantic analysis (ELSA) [152], which is reported

to outperform many other geographical topic-based methods. ELSA is an ESA-based method and is the most similar location-aware news recommendation solution to the work in Chapter 5. But ELSA is different from our work in the following aspects: (i) The proposed LP-ESA in Chapter 5 captures not only the location information but also the personal interests of users for news recommendations, while ELSA is solely based on the location. (ii) Clustering, autoencoders and recommendation-oriented neural networks are used by CLSA, ALSA, and DLSA, respectively, to solve high-dimensionality, sparsity, and redundancy problems in ELSA. (iii) We propose a hybrid model LP-DSA in Chapter 5 to combine and take advantages of DLSA and LP-DSA. Thus, the proposed location-aware recommendation models in Chapter 5 achieve much better performance than ELSA.

ELSA is based on location information only; so, they recommend news solely based on users' locations, but regardless of the users' personal interests (i.e., non-personalized). Thus, different users in the same location will receive the same recommendation, which is unreasonable in practice.

To our knowledge, the apatial topical preference model (STPM) [116] is the only location-aware news recommendation system that considers both the locations and personal interests of users. However, there also exist some differences between STPM and the proposed LP-ESA in Chapter 5. (i) STPM is an LDA-based method, so all textual content in STPM is represented by latent topics, while LP-ESA is ESA-based, so texts are represented with a Wikipedia-based explicit topic space. (ii) More importantly, STPM has a cold start problem: the localized user profiles in STPM are directly learned using the users' history data on the corresponding locations (e.g., home or office), so it fails when users travel to a new place; to avoid this problem, the localized user profiles in LP-ESA are inferred by the users' general profiles (modeled with users' entire his-

tory data) and the local topic distributions of locations; therefore, LP-ESA overcomes the cold start problem and is applicable to recommend news articles at new locations.

2.5 Ontology-Based Personalized Recommendation

An ontology is a formal specification of the types, properties, and interrelationships of the entities for a particular domain of discourse [58]. Since ontologies are usually constructed based on the consensus of domain experts, they are highly reliable and 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], so in order to integrate ontologies from different sources, many metrics have been proposed to quantify the semantic relevance between concepts in the ontologies.

Rada et al. [132] use the length of the shortest path between two concepts to measure their likeness, while the solutions proposed by Wu and Palmer [175] and Ganesan et al. [52] use the relative depths of two concepts and their least common ancestor. Similarly, Jiang and Conrath [76] and Lin [104] propose to use the information content of two compared concepts and their least common ancestor to calculate the similarity between the two concepts. Li et al. [102] propose 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 [66] extend the taxonomic solution and take into account the number of times that the link direction changes, such that the more changes in the directions of relations on the shortest path, the lower the likeness. Differently, our aim in Chapter 6 is not to propose new concept similarity metrics, but to exploit the existing ontology-based concept similarities to propose a novel similarity metric to solve the ambiguity and redundancy problems

in tag-aware personalized recommendation without the need of model training.

Some existing works try to connect folksonomies with ontologies, and propose strategies to solve the co-occurrence problem in the tag-to-ontology allocation. Specifically, Han et al. [61] take the open directory project (ODP) taxonomy[1] as 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: first, the tag allocation is static, i.e., the most appropriate concept is fixed for each tag; second, not all ontologies contain this kind of statistic information as ODP. The use of this method is thus restricted. A more dynamic and adaptive strategy is proposed by Angeletou et al. [9] based on Wu and Palmer's similarity [175]: it first maps the given tag to all possible candidate concepts and then uses the contextual information of its tagset to identify the most relevant concept by computing the similarity of all combinations of tags in this tagset 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 Wu and Palmer's similarity value above a predefined threshold. As an alternative method to adaptively map tags to ontologies, the strategy in [9] will be considered as the competitor of the proposed top-down traversal tag allocation strategy in the evaluation.

There are also some studies that integrate ontologies with folksonomies for personalized search or recommendation. For example, Han et al. [61] investigate how to use domain ontologies to model semantically more intuitive user profiles, and Movahedian and Khayyambashi [114] propose a tag-aware recommender system based on

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

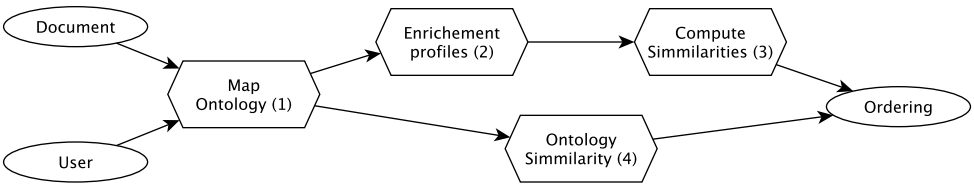


Figure 2.1: Difference between the proposed approach and the state of art

user and document profiles that are both semantically enriched by an external knowledge base. Taxonomies are also leveraged in [71] to generate semantic resource profiles for personalized social tag recommendation. In summary, the purpose of the above works is to solve the *profile modelling problem* in personalized search or recommendation 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 $(1) \rightarrow (2) \rightarrow (3)$, as shown in Figure 2.1. On the other hand, ontologies in the work of Chapter 6 are used to solve the ambiguity and redundancy problems in the social tag space, 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, the personalized ordering procedure in the work of Chapter 6 is $(1) \rightarrow (4)$ in Figure 2.1). Thus, we actually leverage ontological information to solve different problems in personalized search and recommendation, and our proposed techniques can be orthogonally combined with all the above works to obtain a further enhanced tag-aware personalized recommendation (resulting in a hybrid ordering procedure $(1) \rightarrow (2) \rightarrow (1) \rightarrow (4)$).

Furthermore, there are some works to disambiguate tags in folksonomies. Cattuto et al. [38] propose to measure the relatedness between tags using a tag-tag co-

occurrence graph, a FolkRank solution, and three distributional measures with three different vector space representations (tag context, document context, and user context) of tags, where the cosine similarity is used to estimate the similarity between vectors. Consequently, this work 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 [110], where other similarity metrics (e.g., Jaccard and Dice and mutual information) are also introduced. To evaluate the performance of the proposed ontology-based solution in disambiguating tags, these existing approaches will be used as baselines in the experiments in Chapter 6.

Using the ontological similarity for tag disambiguation is also similar to the word sense disambiguation to Wikipedia, or simply disambiguation to Wikipedia (D2W) [133]. 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 [92, 93, 113, 133] have been proposed to utilize 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. [93] also introduce a 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 (seeing articles as concepts, connected by hyperlinks or categories), and then utilizing the uniquely matched co-occurring tags

as context to disambiguate the tags with multiple matching concepts/articles. However, in folksonomies, each user usually only 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 the 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 the experiments have less than 3 uniquely matched tag); so, using only uniquely matched tags as in the D2W works [92, 113, 133] usually can not provide sufficient context for tag disambiguation. Therefore, in Chapter 6, we propose to use all tags that can be matched in the ontology as context to ensure sufficient contextual information for disambiguation. In addition, Wikipedia is not strictly hierarchical; so using Wikipedia as the underlying ontology will make some very effective tree-structure-based concept similarity metrics (e.g., the relative depth concept similarity [52]) and the efficient top-down traversal tag allocation strategy become invalid. Consequently, due to its damage to the recommendation effectiveness and efficiency, Wikipedia is not used as the underlying ontology in our work in Chapter 6.

CHAPTER 3

Preliminaries

3.1 Folksonomy

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. This practice is called *social tagging* or *collaborative tagging*, while the classification systems derived from social tagging activities are known as *folksonomies*. Formally, a *folksonomy* is a tuple $\mathcal{F} = (U, T, I, A)$, where U, T, and I are sets of *users*, *tags*, and *items*, respectively, and $A \subseteq U \times T \times I$ is a set of assignments (u, t, i) of a tag t to an item i by a user u [70]. An example of a folksonomy resulting from the tagging activities in a social tagging system is shown in Figure 3.1, where (i) u_1 uses two tags, *Apache* and *Jaguar*, to annotate the item i_1, and (ii) u_2 uses two tags, *Jaguar* and *Mammal*, to annotate i_2. Given this folksonomy, we have $U = \{u_1, u_2\}$, $T = \{Apache, Jaguar, Mammal\}$, $I = \{i_1, i_2\}$, and $A = \{(u_1, Apache, i_1), (u_1, Jagu-$

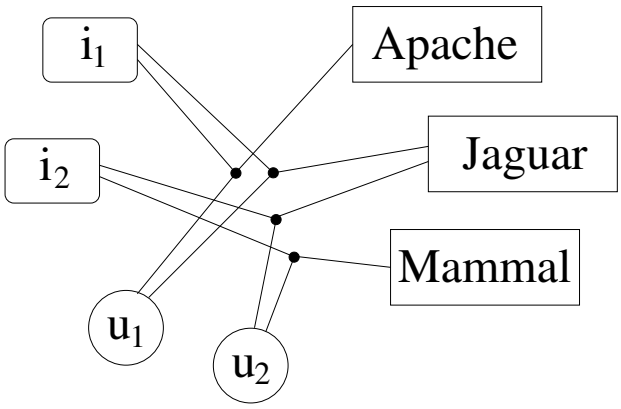


Figure 3.1: Example of a folksonomy

$ar, i_1), (u_2, Jaguar, i_2), (u_2, Mammal, i_2)\}$.

3.2 Vector Space Model

The *vector space model (VSM)* [138] is widely used to model the user and item profiles as weighted vectors of tags. Precisely, a user's profile is usually obtained by aggregating the social tags assigned by this user to all the online items; and it is represented as a weighted vector of tags, where each dimension corresponds to a tag applied by this user, and the weight value of each dimension is influenced by the number of times that the tag is applied by the user. Formally, an *user profile* is a feature vector as follows:

$$x_u = \{g_1^u, g_2^u, \ldots, g_j^u, \ldots, g_M^u\},$$

where $M = |T|$ is the tag vocabulary's size, and $g_j^u = |\{(u, t_j, i) \in A \mid i \in I\}|$ is the number of times that user u annotates items with tag t_j [33].

Similarly, an item's profile is obtained by aggregating the social tags assigned to this item by all users; and it is also represented as a weighted vector, where the weight of each dimension is influenced by the number of times the item is annotated with the

corresponding tag. Formally, an *item profile* is defined as the following vector:

$$x_i = \{g_1^i, g_2^i, \ldots, g_j^i, \ldots, g_M^i\},$$

where $g_j^i = |\{(u, t_j, i) \in A \mid u \in U\}|$ is the number of times that item i is annotated with tag t_j.

Since users can only assign limited number of tags, the values of most dimensions in both user and item profiles are 0. Therefore, to keep it simple and clear for presentation, in this thesis, we omit the dimensions whose corresponding weights are 0; consequently, although the dimensions of all user and item profiles are the same, in this thesis, the user profile is written as

$$x_u = \{(t_{u_1}, w_{u_1}), \ldots, (t_{u_j}, w_{u_j}), \ldots, (t_{u_{m_u}}, w_{u_{m_u}})\},$$

and the item profile is written as

$$x_i = \{(t_{i_1}, w_{i_1}), \ldots, (t_{i_j}, w_{i_j}), \ldots, (t_{i_{m_i}}, w_{i_{m_i}})\},$$

where t_{u_j} (resp., t_{i_j}) is the tag in the user (resp., item) profile, w_{u_j} (resp., w_{i_j}) is the number of times that user u annotates items using tag t_{u_j} (resp., item i is annotated by tag t_{i_j}), and m_u (resp., m_i) is the number of different tags used by u (resp., assigned to i). For example, the user profiles in Figure 3.1 are $x_{u_1} = \{(Apache, 1), (Jaguar, 1)\}$, $x_{u_2} = \{(Jaguar, 1), (Mammal, 1)\}$; while the item profiles are $x_{i_1} = \{(Apache, 1), (Jaguar, 1)\}$, and $x_{i_2} = \{(Jaguar, 1), (Mammal, 1)\}$.

3.3 Personalized Recommendation

Given a user u and a set of items $I = \{i_1, \cdots, i_n\}$, *personalized recommendation* is to produce a ranked recommendation list $\tau = [i_1 \geqslant i_2 \geqslant \cdots \geqslant i_n]$ s.t. $i_a \geqslant i_b$ iff $\mathit{Rlv}(u, i_a) \geqslant \mathit{Rlv}(u, i_b)$, where $\mathit{Rlv}(u, i)$ is a function measuring how relevant an item i is to the user u. In content-based filtering, $\mathit{Rlv}(u, i)$, is usually computed by the similarity between the user profile, x_u, and item profile, x_i. Formally,

$$\mathit{Rlv}(u, i) = \mathit{Sim}(x_u, x_i). \tag{3.1}$$

Furthermore, given a set of news $V = \{v_1, \cdots, v_n\}$ and a user u at location l, *location-aware personalized news recommendation* is to generate a ranked recommendation list $\{v_1 \succeq v_2 \succeq \cdots \succeq v_{|V|}\}$ for all news items, s.t., $v_i \succeq v_j$ iff $R_{u,v_i,l} \geqslant R_{u,v_j,l}$, where $R_{u,v,l}$ is a score measuring the relevance of news v to user u at location l. In content-based filtering, $R_{u,v,l}$ is usually computed by the similarity of the localized user profile $\Phi_{u,l}$ and the localized news profile $\Phi_{v,l}$. Formally,

$$R_{u,v,l} = \mathit{Sim}(\Phi_{u,l}, \Phi_{v,l}). \tag{3.2}$$

3.4 Similarity Measures

Obviously, the performance of personalized recommendation depends greatly on the effectiveness of the adopted similarity measures in $\mathit{Sim}(x_u, x_i)$ and $\mathit{Sim}(\Phi_{u,l}, \Phi_{v,l})$. Therefore, I now briefly discuss two state-of-the-art similarity measures for personalization, namely, *cosine similarity* and *scalar similarity*, which are also used as baselines in the experimental evaluation of Chapter 6.

The most widely adopted similarity measure in content-based personalized rec-

ommendation is the cosine similarity [32, 33, 178]. Following the above VSM approach, the *cosine similarity* between a user profile x_u and a item profile x_i, denoted $\mathit{Sim}_{\mathit{Cosine}}(x_u, x_i)$, is formally defined as follows:

$$\mathit{Sim}_{\mathit{Cosine}}(x_u, x_i) = \frac{x_u \cdot x_i}{\|x_u\|\|x_i\|}, = \frac{\sum_{j=1}^{n}(w_{u_j} \cdot w_{i_j})}{\sqrt{\sum_{j=1}^{n}(w_{u_j})^2} \cdot \sqrt{\sum_{j=1}^{n}(w_{i_j})^2}}, \tag{3.3}$$

where $n = |T|$ is the number of different tags, and every w_{u_j} (resp., w_{i_j}) is the weight of tag t_{u_j} (resp,. t_{i_j}) in profile x_u (resp., x_i).

Later, Vallet et al. [164] proposed the so-called scalar similarity, which is similar to the cosine similarity, except that it eliminates the user and the item profile length normalization factors. In the classic VSM, the finality of the length normalization factor is to penalize the score of items that contain a high amount of information, and might have matched the query only by chance. In terms of a social tagging system, a high amount of related tags is correlated with the popularity of the items among users. Hence, if we used a length normalization factor, we would penalize the score of popular items. As several works point out, this popularity value is a good source of relevancy [14, 70]. Thus, Vallet et al. argue that it would not be advisable to penalize popular items. Formally, the *scalar similarity* between a user profile x_u and a item profile x_i, denoted $\mathit{Sim}_{\mathit{Scalar}}(x_u, x_i)$, is defined as follows:

$$\mathit{Sim}_{\mathit{Scalar}}(x_u, x_i) = x_u \cdot x_i = \sum_{j=1}^{n}(w_{u_j} \cdot w_{i_j}), \tag{3.4}$$

where n and all w_{u_j} and w_{i_j} are defined as above.

ALGORITHM 1: HierarchicalAgglomerativeClustering(T, s, dc)

Input: A set of tags T and a stopping threshold ST
Output: a set of clusters

```
1 Assign each tag to its own singleton cluster;
2 do
3 |   Computer the similarities between remaining clusters;
4 |   join the two clusters with the highest similarity value;
5 while number of remaining clusters is larger than ST;
6 return the resulting set of clusters
```

3.5 Clustering

Clustering is to group a set of objects such that objects in the same group (called *cluster*) are more similar (in some sense) to each other than to those in other groups (clusters). There are many kinds of clustering models, such as hierarchical clustering and k-means clustering. The work in [148] has proven that hierarchical clustering achieves much better performance than k-means clustering in the personalized recommendation using social annotations. Therefore, hierarchical clustering is used in this work.

Hierarchical clustering is a kind of clustering model, aiming to build a hierarchy of clusters. There are generally two strategies for hierarchical clustering: agglomerative (bottom-up) and divisive (top-down), where agglomerative strategy is more commonly used because of its computation efficiency.

The process of *hierarchical agglomerative clustering* is shown in Algorithm 1. The input tags are represented as a weight vector over the set of the corresponding resources (e.g., items in Chapter 4 and news in Chapter 5). The similarity between two clusters A and B is computed as follows.

$$Sim_{clustering}(A, B) = \frac{\sum_{a \in A} \sum_{b \in B} Sim(a, b)}{|A| \cdot |B|}, \tag{3.5}$$

where a and b are tags in clusters A and B, respectively; $|\cdot|$ is size of a cluster.

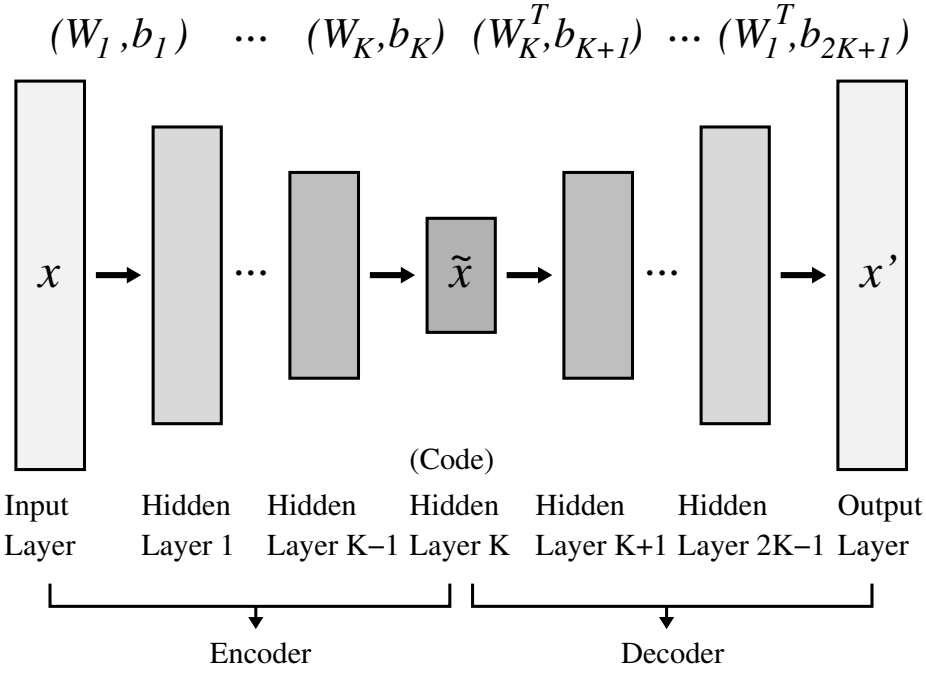


Figure 3.2: Overview of an autoencoder

3.6 Autoencoder

An *autoencoder* is a neural network used to learn a feature representation (also called *encoding*) for a set of data, which is commonly used for dimensionality reduction.

As shown in Figure 3.2, an autoencoder is a feed-forward neural network, consisting of two parts: *encoder* and *decoder*. Generally, the encoder takes inputs x at the input layer and then passes them through multiple hidden layers and projected to abstract feature representations $\widetilde{x}_u$ at the code layer; then, $\widetilde{x}_u$ is used as inputs of the decoder to generate reconstructed representation of x (denoted x') at the output layer. The training of an autoencoder is to minimize the reconstruction errors which are computed as the Euclidean (i.e., L2) norms of the differences between x and x'; formally,

$$L_{autoencoder}(\Theta) = \|x - x'\|. \tag{3.6}$$

Specifically, given a weight matrix W_1, and a bias vector b_1, the intermediate output h_1 of the first hidden layer at the encoder is formally defined as follows:

$$h_1(x) = \sigma(W_1 x + b_1), \tag{3.7}$$

where σ is the selected activation function, where sigmoid function or hyperbolic tangent (denoted $\tanh$) function is commonly used.

Similarly, the intermediate output of the jth hidden layer h_j, $j \in \{2, \ldots, K\}$ at the encoder, is defined as:

$$h_j(x) = \sigma(W_j h_{j-1}(x) + b_j), \tag{3.8}$$

where W_j and b_j are the weight matrix and the bias vector for the jth hidden layer, and K is the total number of hidden layers. The outputs of the Kth hidden layer are the abstract feature representations $\widetilde{x}$. Formally,

$$\widetilde{x} = h_K(x). \tag{3.9}$$

As for the decoders, since a tied-weight strategy is adopted, the weight matrices in the decoder are the transposes of those in the encoder. Therefore, the intermediate output of the $K+j$th hidden layer h_{K+j}, $j \in \{1, \ldots, K-1\}$ at the decoder, is formally

defined as:

$$h_{K+j}(x) = \sigma(W_{K-(j-1)}^{T} h_{K+(j-1)}(x) + b_{K+j}) \tag{3.10}$$

where $W_{K-(j-1)}^{T}$ is the transpose of $W_{K-(j-1)}$, and b_{K+j} is the bias vector for the $K + j$th hidden layer. The outputs of the $2K - 1$th hidden layer are used to generate reconstructed representations x' in the output layer:

$$x' = \sigma(W_1^T h_{2K-1}(x) + b_{2K}). \tag{3.11}$$

CHAPTER 4

Deep Tag-Aware Personalized Recommender Systems

Despite the huge prevalence of social tags, an outstanding problem of using tags for personalized recommendation is the *uncontrolled vocabulary problem*: as users can freely choose their own vocabulary, social tags may contain many uncontrolled vocabularies, such as homonyms, synonyms, words in arbitrary languages, or even user-created words. This usually results in very sparse, redundant, and ambiguous tag information, which greatly degrades the performance of tag-aware recommendation systems.

A solution to this problem is to apply clustering in the tag space [148]; however, clustering requests to compute the similarity between tags, which is usually very time-consuming. Another solution is to use autoencoders; in [203], abstract feature representations for tag-based user profiles are first modeled by autoencoders and then used

as inputs of user-based collaborative filtering to generate recommendations. Although this method is reported to achieve better performance than the clustering-based collaborative filtering method [203], it still suffers from the following drawback: the model's learning signal is not directly related to the objective of personalized recommendation, i.e., distinguishing the user's target items from the irrelevant ones; so, the resulting abstract feature representations are not very effective for personalized recommendation. Consequently, its improvement in personalized recommendation is not very significant.

Motivated by the above observations, in this chapter, we address the uncontrolled vocabulary problem by using deep neural networks to map the tag-based user and item profiles to an abstract deep feature space, where the similarities between users and their target items (resp., irrelevant items) are maximized (resp., minimized). We call the similarities on the deep feature space *deep-semantic similarities* and this model the *deep-semantic similarity-based personalized recommendation* (DSPR) model.

Despite achieving superior performance, the training of the DSPR model is usually very time-consuming in practice, mainly because DSPR has many hidden layers and a huge number of candidate items, which result in the *vanishing gradient problem* and *scale problem*. Therefore, we further propose to solve these problems by *hybrid deep learning* and *negative sampling*, respectively. With both improvements, the resulting hybrid deep learning-based personalized recommendation with negative sampling (HDLPR-NS) model becomes very efficient in model training (hundreds of times quicker than DSPR), whilst maintaining superior recommendation performance.

The rest of this chapter is organized as follows. The DSPR model is proposed in Section 4.1. Section 4.2 further proposes HDLPR-NS to uses hybrid deep learning and negative sampling to enhance the deep model's training efficiency. Finally, experimental results are presented and evaluated in Section 4.3.

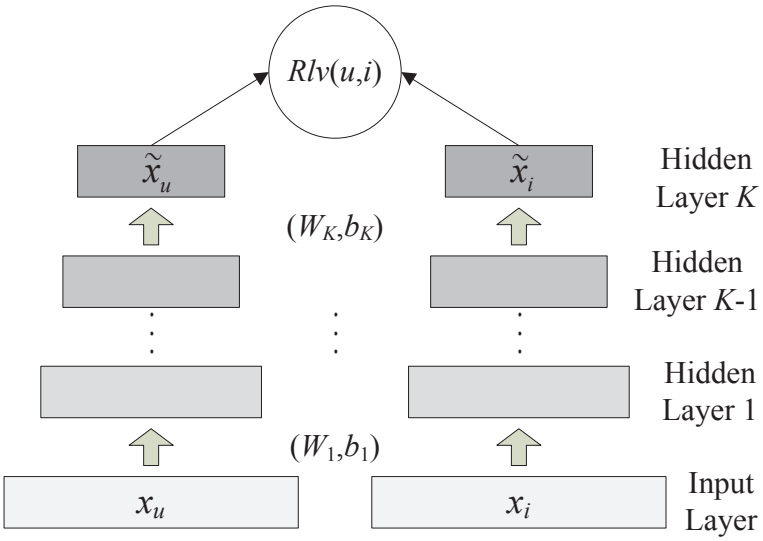


Figure 4.1: Overview of DSPR

4.1 Deep-Semantic Similarity-Based Personalized Recommendation

Figure 4.1 shows an overview of the deep-semantic similarity-based personalized recommendation (DSPR) model. Generally, DSPR takes the tag-based user and item profiles x_u and x_i (as defined above) as inputs of two deep neural networks with shared parameters. These inputs are then passed through multiple hidden layers and projected into an abstract deep feature space on the final hidden layer, where the similarities between the abstract representations of user and item profiles are computed. Finally, ranked recommendation lists are generated using relevance scores, computed by applying the softmax function on the resulting similarities.

The motivation behind sharing parameters in DSPR is as follows: (i) As in all the existing content-based tag-aware recommender systems [33, 148, 203], users and items in DSPR are described using the same set of tags from the same folksonomy; so,

the user and item profiles share the same tag space and the same value range in each dimension, and it is thus reasonable to use shared parameters to reduce computational and memory costs in model training. (ii) Shared parameters ensure that the two neural networks share an abstract feature space in the final layer, where users and items are represented using the same features, which is beneficial for measuring their relevance in content-based recommendations.

Formally, given the user profile x_u, the item profile x_i, a weight matrix W_1, and a bias vector b_1, the intermediate output h_1 of the first hidden layer is defined as follows:

$$h_1(u) = \tanh(W_1 x_u + b_1), \tag{4.1}$$

$$h_1(i) = \tanh(W_1 x_i + b_1), \tag{4.2}$$

where $\tanh$ is used as the activation function. Similarly, the intermediate output of the jth hidden layer h_j, $j \in \{2, \ldots, K\}$, is defined as:

$$h_j(u) = \tanh(W_j h_{j-1}(u) + b_j), \tag{4.3}$$

$$h_j(i) = \tanh(W_j h_{j-1}(i) + b_j), \tag{4.4}$$

where W_j and b_j are the weight matrix and the bias vector for the jth hidden layer, and K is the total number of hidden layers. The outputs of the Kth hidden layer are the abstract feature representations of user and item profiles, denoted $\widetilde{x}_u$ and $\widetilde{x}_i$, respectively. Formally,

$$\widetilde{x}_u = h_K(u), \qquad \widetilde{x}_i = h_K(i). \tag{4.5}$$

Then, the similarity between a user u and an item i is measured using the cosine similarity between the abstract representations of their profiles, formally defined as

$$Sim(u, i) = \frac{\widetilde{x}_u \cdot \widetilde{x}_i}{\|\widetilde{x}_u\| \|\widetilde{x}_i\|}, \tag{4.6}$$

and called *deep-semantic similarity*.

Finally, the relevance scores [72] of items i to a given user u are measured by applying the softmax function on the resulting deep-semantic similarities between u and i, which are then used to rank a personalized recommendation list for the given user u. Formally,

$$Rlv(u, i) = \frac{e^{Sim(u,i)}}{\sum_{i' \in I} e^{Sim(u,i')}}. \tag{4.7}$$

Given an assumption that the target items of a given user are those annotated by this user, to achieve good personalized recommendation, these items should have higher relevance scores than others. We thus conduct the model training with an objective to maximize the relevance scores of target items; equivalently, this means to maximize the deep-semantic similarities between users and their target items and minimize those

with irrelevant ones. Formally, this is equivalent to minimizing the following loss function:

$$\begin{aligned} L(\Theta) &= -\sum_{(u,i^*)} \log(Rlv(u,i^*)) \\ &= -\sum_{(u,i^*)} \left[\log(e^{Sim(u,i^*)}) - \log(\sum_{i' \in I} e^{Sim(u,i')}) \right], \end{aligned} \tag{4.8}$$

where Θ represents the parameters W_j and b_j in the neural networks; (u, i^*) are training samples, which are pairs of a user u and his/her target item i^*, generated from assignments (u, t, i^*) in a training dataset.

In training, we first initialize the weight matrices W_j, using the random normal distribution, and initialize the biases b_j to be zero vectors; the model is then trained by back-propagation using mini-batch gradient descent; finally, the training stops when the model converges or reaches the maximum training runs.

4.2 Hybrid Deep Learning-Based Personalized Recommendation with Negative Sampling

DSPR's learning signal is directly correlated with differentiating the user's target items from the irrelevant ones, so it achieves superior performance in personalized recommendation. However, since there are many hidden layers in deep neural networks and a huge number of candidate items, the training of DSPR is usually very time-consuming in the real-world context. Therefore, to enhance DSPR's training efficiency and ensure the scalability in practice, we propose to use *hybrid deep learning* and *negative sam-*

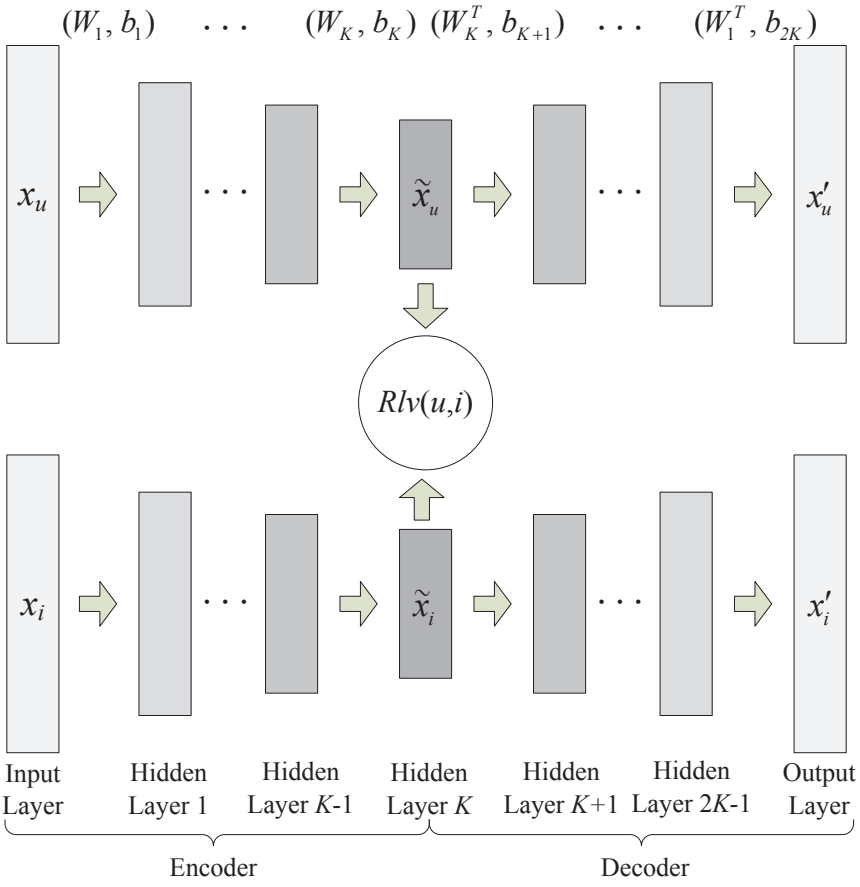


Figure 4.2: Overview of HDLPR

pling [112] to reduce the number of training runs needed for model convergence and to reduce the processing time needed for each training run, respectively. The resulting efficient model is called *hybrid deep learning-based personalized recommendation with negative sampling* (HDLPR-NS).

4.2.1 Hybrid Deep Learning

Training deep neural networks using back-propagation is difficult, because the learning signals become minuscule and insignificant when they are backpropagated to the

first few layers, which results into a very slow learning progress and numerous training runs for model convergence. This problem is called *diffusion or vanishing of gradients*, which can be remedied by pre-training each layer before using back-propagation to fine-tune the entire network [18, 65]. In this work, we propose a new solution, called *hybrid deep learning*, which is based on the idea of integrating autoencoders with the neural networks of DSPR to generate additional learning signals based on *reconstruction errors*, which is then combined with the deep-semantic similarity-based relevance scores in DSPR to form a hybrid deep learning signal for model training. The model is called *hybrid deep learning-based personalized recommendation (HDLPR).*

Intuitively, for the following reason, adding reconstruction errors as learning signals can greatly accelerate the deep model's learning progress and reduce training runs needed for model convergence: In DSPR, the deep-semantic similarity-based learning signal becomes very weak when it is back-propagated to the first few layers, so learning the first few weight matrices, e.g., W_1 and W_2, is very slow. In HDLPR, since *tied weights* are used in autoencoders, i.e., the weight matrices in the decoder are the transposes of those in the encoder, the reconstruction-error-based learning signal will be used to first update W_1^T, then back-propagated to updating W_2^T, W_3^T, and so on. As updating W_j^T is equivalent to updating W_j, it remedies the diffusion of gradients in DSPR.

Figure 4.2 shows the overall process of the proposed hybrid deep learning-based personalized recommendation (HDLPR) model. The structure of HDLPR is similar to the one of DSPR, but adds K layers to form a decoder; so, by taking the first $K+1$ layers as encoder, we convert each neural network to an autoencoder with tied weights. Then, the decoders take the abstract feature representations of user and item profiles, $\widetilde{x}_u$ and $\widetilde{x}_i$, as input and generate reconstructed user and item profiles in their output layers,

denoted x'_u and x'_i. Finally, the reconstruction errors of user (resp., item) profiles are computed as the Euclidean (i.e., L2) norms of the differences between x_u (resp., x_i) and x'_u (resp., x'_i).

Formally, the definitions of layers in encoders are the same as the ones in the neural networks of DSPR. As for the decoders, the intermediate output of the $K + j$th hidden layer h_{K+j}, $j \in \{1, \ldots, K-1\}$, is formally defined as:

$$h_{K+j}(u) = \tanh(W^T_{K-(j-1)} h_{K+(j-1)}(u) + b_{K+j}), \tag{4.9}$$

$$h_{K+j}(i) = \tanh(W^T_{K-(j-1)} h_{K+(j-1)}(i) + b_{K+j}), \tag{4.10}$$

where $W^T_{K-(j-1)}$ is the transpose of $W_{K-(j-1)}$, and b_{K+j} is the bias vector for the $K + j$th hidden layer. The outputs of the $2K - 1$th hidden layer are used to generate reconstructed user and item profiles, denoted x'_u and x'_i, in the output layer:

$$x'_u = \tanh(W_1^T h_{2K-1}(u) + b_{2K}), \tag{4.11}$$

$$x'_i = \tanh(W_1^T h_{2K-1}(i) + b_{2K}), \tag{4.12}$$

Then, the reconstruction errors of user (resp.,item) profiles are computed as the Euclidean (i.e., L2) norms of the differences between x_u (resp., x_i) and x'_u (resp., x'_i). By integrating the reconstruction errors with the deep-semantic similarity-based relevance scores in DSPR, the training objective of HDLPR is to maximize the relevance scores of target items and also to minimize the reconstruction errors of the user and target item

profiles in each training sample. Formally, it is equivalent to minimize the following hybrid loss function.

$$
\begin{aligned}
L_h(\Theta) = \lambda_\theta & \left(\sum_{j=1}^{K} \|W_j\|^2 + \sum_{j=1}^{2K} \|b_j\|^2 \right) \\
& + \lambda_e \sum_{(u,i^*)} (\|x'_u - x_u\| + \|x'_{i^*} - x_{i^*}\|) \\
& - (1 - \lambda_\theta - \lambda_e) \sum_{(u,i^*)} \left[\log(e^{Sim(u,i^*)}) - \log(\sum_{i' \in I} e^{Sim(u,i')}) \right],
\end{aligned} \tag{4.13}
$$

where the first term is a L2 regularization used to prevent overfitting; the second term is the sum of reconstruction errors of the user and target item profiles in each training sample; and the third term is the deep-semantic similarity-based learning signal, $L(\Theta)$, as defined in Equation (4.8); λ_θ and λ_e are parameters representing different importances of the corresponding terms, where $\lambda_e > 0$, $\lambda_\theta > 0$, and $\lambda_\theta + \lambda_e < 1$.

4.2.2 Negative Sampling

Although hybrid deep learning greatly reduces the number of training runs needed for model convergence, the processing of each training run in both DSPR and HDLPR is still very time-consuming, since the deep-semantic similarity-based loss function $L(\Theta)$ as defined in Equation (4.8), which is also the third term of $L_h(\Theta)$ in Equation (4.13), is computationally very expensive.

Specifically, for each training sample (u, i^*) in each training run, the second term of $L(\Theta)$ requests to compute and sum the deep-semantic similarities between u and all candidate items in I. In practice, the number of candidate items for an online recom-

mendation system is usually very large (millions), and training a deep neural network often requires numerous training samples; therefore, the cost of processing each training run in both DSPR and HDLPR is very high. However, this term is essential: with its help, minimizing $L(\Theta)$ not only maximizes the deep-semantic similarity between the given user and his/her target items, but also minimizes those with irrelevant items. Consequently, it helps to distinguish the target item from the irrelevant ones.

To tackle this dilemma, in this work, we use *negative sampling* [112] to greatly reduce the time needed to process each training sample. In negative sampling, instead of using all irrelevant items, for each training sample, we randomly sample only a small number (S) of irrelevant items from the set of candidate items as *negative examples* to approximate the noise of irrelevant items and to differentiate target items from irrelevant ones. The resulting efficient model is called HDLPR with negative sampling (HDLPR-NS); and its loss function is formally defined as follows:

$$
\begin{aligned}
L_h^{NS}(\Theta) = \lambda_\theta & \left(\sum_{j=1}^{K} \|W_j\|^2 + \sum_{j=1}^{2K} \|b_j\|^2 \right) \\
& + \lambda_e \sum_{(u,i^*)} (\|x'_u - x_u\| + \|x'_{i^*} - x_{i^*}\|) \\
& - (1 - \lambda_\theta - \lambda_e) \sum_{(u,i^*)} \left[\log(e^{Sim(u,i^*)}) - \log(\sum_{(u,i^-)\in D^-} e^{Sim(u,i^-)}) \right], \qquad (4.14)
\end{aligned}
$$

where (u, i^-) are negative samples, which are contained in a negative dataset D^- and generated by randomly sampling S negative examples i^- for each training sample (u, i^*).

4.3 Experiments

To show the strength of the proposed DSPR and HDLPR-NS models in solving the uncontrolled vocabulary problem and offering superior personalized recommendation performance, we use the following three models based on state-of-the-art solutions for the same problem, clustering [148] and autoencoders [203], as baselines:

- Clustering-based cosine similarity (**CCS**): hierarchical clustering [148] is applied to model the users and items as cluster-based feature vectors, upon which content-based filtering using cosine similarity is applied for recommendations.
- Clustering-based collaborative filtering (**CCF**): CCF is similar to CCS in feature modeling, but applies user-based collaborative filtering for recommendations.
- Autoencoder-based collaborative filtering (**ACF**) [203]: an autoencoder is used to obtain abstract representations of user profiles, upon which user-based collaborative filtering is applied for recommendations.

For a fair comparison, the experiments are performed on the same public real-world datasets, Delicious and Last.Fm, as used in [203], which are gathered from the Delicious bookmarking system and the Last.Fm online music system, respectively, and are both released in HetRec 2011 [34]. After using the same pre-processing as in [203] to remove infrequent tags, which are used less than 15 times in the Delicious dataset and less than 5 times in the Last.Fm dataset, the statistic information of the resulting datasets are as shown in Table 4.1. Both the original datasets[1] and pre-processed datasets[2] are online available for research uses.

[1] http://ir.ii.uam.es/hetrec2011/datasets.html
[2] http://goo.gl/iQDRBo

Table 4.1: Dataset information

Dataset	Users	Tags	Items	Assignments
Delicious	1 843	3 508	65 877	339 744
Last.Fm	1 808	2 305	12 212	175 641

Experiments are conducted using five-fold cross validation, where both datasets are randomly partitioned into five equal-size subsets. Of the five subsets, a single subset is retained for validation and testing, while the remaining four subsets are used as training set. The cross-validation process is then repeated five times, with each of the five subsets used exactly once as the validation and test set. The results from the five different folds are then averaged as the final results. In this work, in order to avoid over-fitting by early stopping, we further randomly partition the validation and test set: one quarter of the set is used for validation and the rest is used for testing. Consequently, in each fold, the whole dataset are randomly divided into a training set (80% of total assignments), a validation set (5% of total assignments), and a test set (15% of total assignments).

Furthermore, given an assumption that the target items of a given user are those annotated by this user, the assignments (u, t, i^*) in the training set are used to construct user and item profiles and to extract the user-item pairs (u, i^*) as training samples. We also extract user-item pairs from the assignments in the validation set as validation samples, which are used to avoid over-fitting by early stopping. Finally, user-item pairs extracted from the assignments in the test set are used as test samples to evaluate the recommendation performance.

All models are implemented using Python and Theano, and run on a GPU server with an NVIDIA Tesla K40 GPU and 12GB GPU memory. The parameters of DSPR are set as follows: (i) the number of hidden layers (i.e., K) is 3; (ii) the number of

neurons in the first, second, and third hidden layers are $2\,000$, 300, and 128, respectively; (iii) the training batch size is 128; and (iv) the learning rate for model training is $\eta = 0.005$. These parameters are the same in HDLPR-NS with the following additional settings: (v) adding two hidden layers and the number of of neurons in the fourth and fifth hidden layers are 300 and $2\,000$, respectively; (vi) the balancing parameters λ_θ and λ_e are set to 0.01 and 0.2, respectively; (vii) the number of negative examples for each training sample (i.e., S) is 127.

Some of the above parameters are set empirically, e.g., the numbers of neurons in hidden layers and the training batch size. For other parameters, such as the learning rate η and the balancing parameters λ_θ and λ_e, we determine their values based on experiments using grid search: We first iteratively select the value of λ_θ from 0 to 0.05 with a step of 0.01, select the value of λ_e from 0.1 to 0.5 with a step of 0.1, and select the value of η from a set of values $\{0.0005, 0.001, 0.005, 0.01\}$; then, the five-fold cross validation is conducted and the combination with the best recommendation performance is used.

The most popular metrics for the evaluation of recommendation systems are precision, recall, and F1-score [27]. Since users usually only browse the topmost recommended items, 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 *F*1*-score at k ($F@k$)*. Formally,

$$P@k = \frac{1}{|U'|} \sum_{u \in U'} P_u@k, \qquad P_u@k = \frac{C_u@k}{k}, \tag{4.15}$$

$$R@k = \frac{1}{|U'|} \sum_{u \in U'} R_u@k, \qquad R_u@k = \frac{C_u@k}{N_u}, \tag{4.16}$$

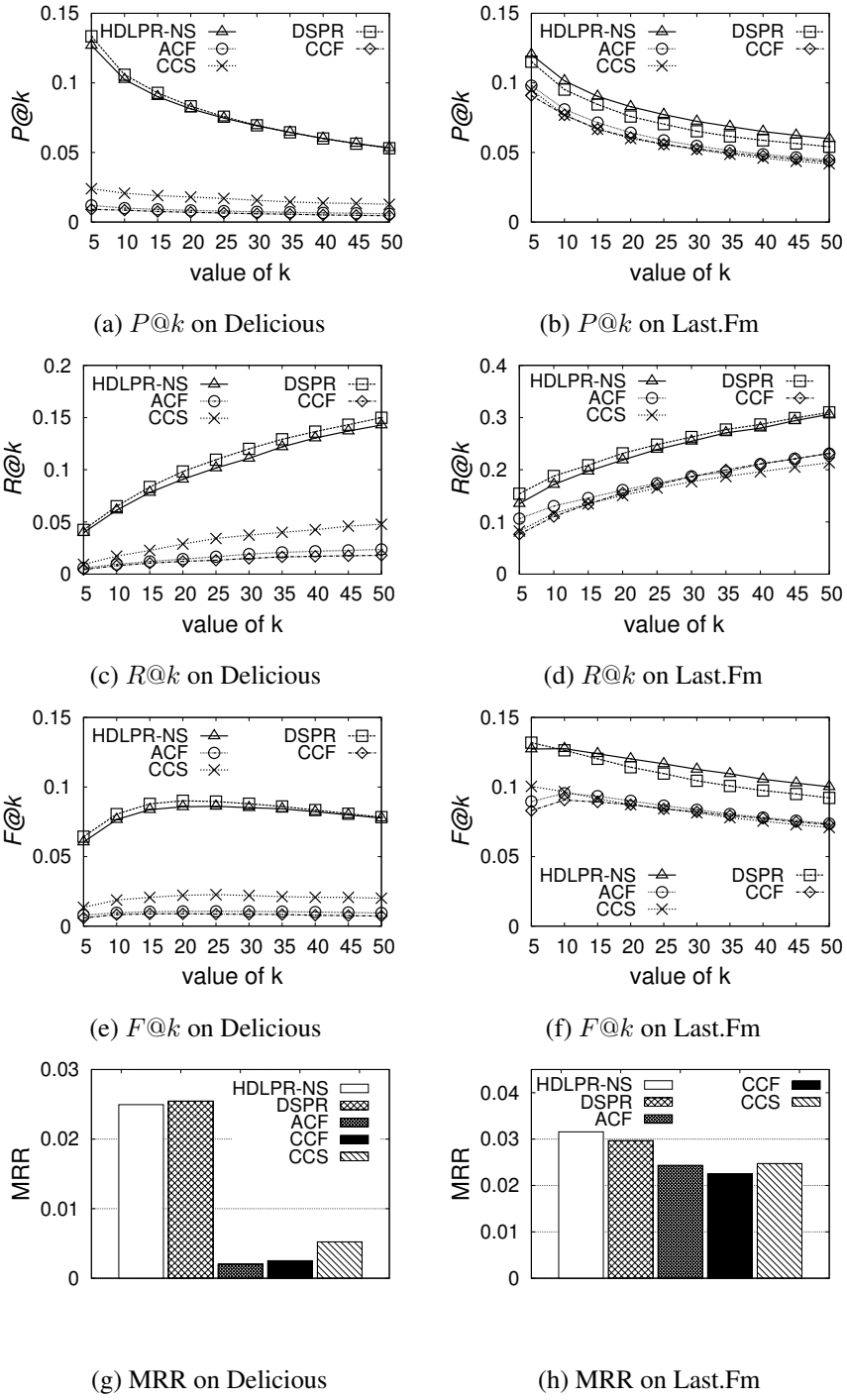
$$F@k = \frac{2 \cdot P@k \cdot R@k}{P@k + R@k}, \tag{4.17}$$

where k is the length of recommendation list; U' is the set of users in the test 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 the given user u's target items in the recommendation list with cut-off rank k; and N_u is the total number of u's target items in test sets.

Since users always prefer to have their target items ranked in the front of the recommendation list, we also employ the *mean reciprocal rank (MRR)* [166] as evaluation metric, which gives greater importance to items ranked higher. MRR measures the performance of a personalized function by assigning a value $1/r$ for each test sample (u, i^*) and then computing the mean value. Formally,

$$MRR = \frac{1}{n} \sum_{j=1}^{n} \frac{1}{r_j}, \tag{4.18}$$

where r_j is the ranking position of the target item of the jth test sample in the personalized recommendation list, and n is the total number of test samples.



(a) $P@k$ on Delicious

(b) $P@k$ on Last.Fm

(c) $R@k$ on Delicious

(d) $R@k$ on Last.Fm

(e) $F@k$ on Delicious

(f) $F@k$ on Last.Fm

(g) MRR on Delicious

(h) MRR on Last.Fm

Figure 4.3: The personalized recommendation performances in terms of $P@k$, $R@k$, $F@k$, and MRR

4.3.1 Main Results

Figure 4.3 depicts in detail the personalized recommendation performances of DSPR, HDLPR-NS, and three baselines on the Delicious and Last.Fm datasets in terms of MRR, $P@k$, $R@k$, and $F@k$, where ten cut-off ranks $k = 5, 10, \ldots, 50$ are selected for evaluation.

As shown in Figure 4.3, despite some slight differences due to the random weights' initializations and dataset partitions, the performance results of CCF and ACF in Figure 4.3 are highly consistent with the results reported in [203] in terms of orders of magnitude, tendencies, and relative performances. Furthermore, the performances of the proposed DSPR and HDLPR-NS models are very similar on the Delicious dataset, while HDLPR-NS slightly outperforms DSPR on the Last.Fm dataset. This indicates that integrating hybrid deep learning and negative sampling with DSPR will not degrade its performance (and sometimes even improve it).

In general, Figure 4.3 shows that the DSPR and HDLPR-NS models both significantly outperform the three baselines in all metrics, e.g., the MRRs of DSPR and HDLPR-NS are both more than 3.8 times (resp., 20%) better than the best baseline, CCS, on Delicious (resp., Last.Fm), and the average improvements in $P@k, R@k$, and $F@k$ are also similar. The superior performances of DSPR and HDLPR-NS are mainly because the training objectives of DSPR and HDLPR-NS are both directly correlated with distinguishing the user's target items from the irrelevant ones; so, the resulting abstract features for user and item profiles are much more effective representations for personalized recommendation than those generated by clustering and autoencoders.

We also note in Figure 4.3 that DSPR and HDLPR-NS achieve much higher improvements on Delicious than on Last.Fm, as DSPR and HDLPR-NS maintain a relatively stable performance on both datasets, but the performance of the baselines dra-

Table 4.2: DSPR vs. HDLPR-NS in training time and quality

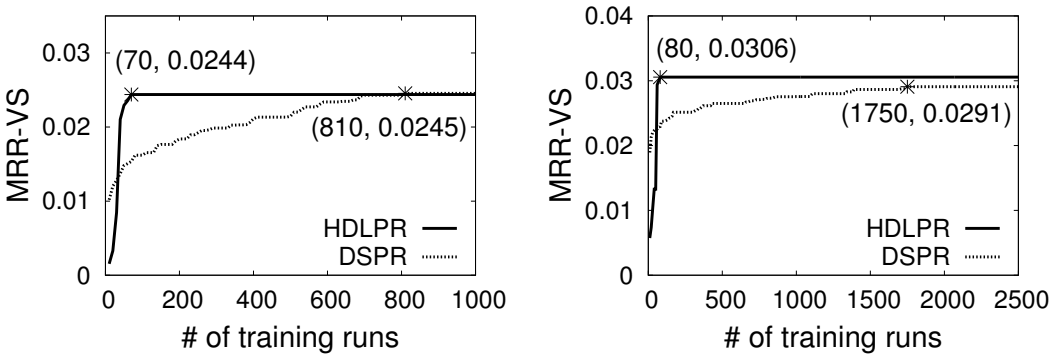
	Delicious		Last.Fm	
	DSPR	HDLPR-NS	DSPR	HDLPR-NS
# of runs to converge	810	70	1750	80
MRR-VS	0.0245	0.0243	0.0291	0.0304
time for one run (hrs)	0.405	**0.0095**	0.1133	**0.0037**
total time-cost (hrs)	328.0	**0.665**	198.3	**0.296**

matically degrades on Delicious: all metrics are $1/5$ or less than those on Last.Fm. This may be because items in Last.Fm are solely in the domain of music, but items in Delicious are webpages in various domains; so, even with higher tag-removing threshold, Delicious still has a much more redundant, and ambiguous tag space than Last.Fm, which can not be properly handled by the baselines. This shows that DSPR and HDLPR-NS can solve the uncontrolled vocabulary problem better than the baselines; and the more uncontrolled the dataset, the higher the achieved improvement.

4.3.2 Efficiency and Scalability

We also investigate the training efficiency and scalability of DSPR and HDLPR-NS. The training time is recorded to compare the training efficiency of the two models. But the standard training loss is not suitable to compare the models' training quality, as DSPR and HDLPR-NS have different loss functions. Here, we use MRR on validation samples (MRR-VS) to measure the training quality, because (i) the training objectives of both models are to get better performance in personalized recommendation, so the higher the MRR-VS the better the models, and (ii) the values of MRR-VS are computed every 10 training runs to avoid over-fitting, so using it will not increase the training time.

We first explore the effect of using hybrid deep learning. To avoid interference, we



(a) Delicious (b) Last.Fm

Figure 4.4: Training progress in DSPR and HDLPR

use HDLPR (hybrid model without negative sampling) for comparison. As shown in Figure 4.4, with the increase of the number of training runs, HDLPR converges much faster than DSPR on both datasets. DSPR takes 810 (resp., 1750) training runs to reach model convergence on Delicious (resp., Last.Fm), which costs HDLPR only 70 (resp., 80) runs. This shows that hybrid deep learning can significantly enhance the model's learning progress and reduce the training runs needed for model convergence by tens of times. Also, when the models converge, the MRR-VS of HDLPR is almost the same as that of DSPR on Delicious (0.0244 in HDLPR vs. 0.0245 in DSPR) and is slightly higher on Last.Fm (0.0306 in HDLPR vs. 0.0291 in DSPR), which demonstrates that using hybrid deep learning will not degrade the models' training quality.

We then apply negative sampling on HDLPR and find that the resulting HDLPR-NS has very similar training progress to HDLPR: as shown in Table 4.2, it takes HDLPR-NS 70 (resp., 80) runs to converge at MRR-VS $= 0.0243$ (resp., $= 0.0304$) on Delicious (resp., Last.Fm); these values are almost the same as those of HDLPR as shown in Figure 4.4. Therefore, using negative sampling will not influence the benefit of using hybrid deep learning. Furthermore, Table 4.2 also shows that, with the help of negative sampling, the time needed to process one training run on Delicious (resp., Last.Fm)

is greatly reduced from 0.405 (resp., 0.1133) hours in DSPR to 0.0095 (resp., 0.0037) hours in HDLPR-NS. This proves that using negative sampling can substantially reduce the processing time needed for each training run.

Consequently, the total time-cost to reach model convergence in HDLPR-NS is only 0.665 (resp., 0.296) hours on Delicious (resp., Last.Fm), which is roughly 492 (resp., 668) times quicker than that in DSPR. In summary, by using hybrid deep learning and negative sampling, the model training efficiency of HDLPR-NS is hundreds of times better than the one of DSPR, while maintaining similar training quality.

CHAPTER 5

Location-Aware Personalized News Recommender Systems

As presented in Chapter 1, the state-of-the-art geographical-topic-based location-aware new recommendation model, ELSA, suffers from two drawbacks:

- In ELSA, news and locations are projected to a Wikipedia-based topic space, where each Wikipedia concept is regarded as a possible topic. Due to the huge volume (millions) of concepts on Wikipedia, the Wikipedia-based topic space in ELSA is very high dimensional, sparse, and redundant. The high dimensionality problem thus reduces the efficiency of ELSA-based recommendation models, while the sparsity and redundancy degrades the recommendation accuracy.
- ELSA takes into account only the users' geographical information for news recommendation and ignores their personal news interests. This is, however, un-

reasonable for real-world situations. In fact, users' real-time news preferences are influenced by both their long-term personal interests and their real-time geographical context. For example, a housewife may prefer to read entertainment news at home, while the husband may like sports news at the same place. This difference comes from their long-term personal interests, which thus should also be considered in location-aware news recommendation.

Therefore, motivated by these observations, we propose to overcome the drawbacks of ELSA and to further improve the performance of location-aware news recommender systems by the following solutions: We first propose to apply machine learning techniques to resolve high-dimensionality, sparsity and redundancy problems in the Wikipedia-based topic space in ELSA. We then extend ELSA to also take into account the user's long-term personal interests to achieve better personalized news recommendation. Finally, a hybrid location-aware personalized news recommendation model that combines and has the advantages of the above two improvements are proposed.

Specifically, in order to solve high-dimensionality, sparsity and redundancy problems, we propose three novel geographical topic feature models, *clustering-based localized semantic analysis (CLSA)*, *autoencoder-based localized semantic analysis (ALSA)* and *deep localized semantic analysis (DLSA)* to achieve better recommendation performance by *topic feature modelling*. Generally, CLSA, ALSA, and DLSA integrate clustering, autoencoders, and recommendation-oriented deep neural networks respectively, with ELSA to extract denser, more abstract and lower dimensional topic features from the Wikipedia-based topic space in ELSA for the representations of news and locations. The experimental studies show that CLSA, ALSA, and DLSA all improve the performance of ELSA in location-aware news recommendations; while DLSA achieves the best results. This is consistent with the previous findings of the

superior performance of recommendation-oriented neural networks in deep tag-aware personalized recommendation in Chapter 4.

In addition, we also propose to take into account the users' long-term interests to extend the geographical topic feature model, ELSA, to a *location-aware personalised news recommendation with explicit semantic analysis (LP-ESA)* model. LP-ESA offers news recommendation based not only on the user's real-time location information, but also on his/her long-term personal interests, where a collection of user history data (e.g., query history, browsing history, or tweeting history) is used to model the given user's personal interests, and collections of geo-tagged documents is used as descriptions of locations.

Precisely, similarly to ELSA, LP-ESA first projects all the textual items (i.e., geo-tagged documents, user history data, and news articles) onto a Wikipedia-concept-based topic space using ESA [51]. Consequently, the users' interests and news articles are both modeled as weighted topic vectors and called *general user profiles* and *general news profiles*, respectively; while *local topic distributions* for locations are obtained by considering both the topic vectors of these locations and link information between the corresponding topics (Wikipedia concepts). Then, for each given user at a location, based on the general user profile and local topic distribution, LP-ESA constructs a *localized user profile* representing the probabilities of this user's news preferences (in terms of topics) at this location. Similarly, a *localized news profile* is also constructed for each news at a given location indicating the distribution of topics in the news at this location. Finally, the relevance of a news article to a user at a given location is measured by the similarity of this user's localized user profile and the localized news profile; and the news with top-k highest relevance scores are recommended. As LP-ESA utilizes both the geographical and preference information of users, it can achieve

a much better news recommendation performance than ELSA.

Nevertheless, since LP-ESA is also a ESA-based method, similar to ELSA, it also suffers from high-dimensionality, sparsity, and redundancy problems. Therefore, we further integrate the intuitions of LP-ESA and DLSA to also exploit recommendation-oriented deep neural networks to map the (Wikipedia-concept-based) topic space in LP-ESA to an abstract, dense, and low dimensional feature space, where the localized similarities between the users and their target news are maximized, and those with ir-relevant news are minimized. This hybrid model is called *location-aware personalized news recommendation with deep semantic analysis (LP-DSA)*, which is a deep learning extension of LP-ESA as well as a personalized extension of DLSA.

The rest of this chapter is organized as follows. I first briefly introduce the ELSA method, which is the most closely related state-of-the-art geographical topic-based location-aware news recommendation method, in Section 5.1. The proposed CLSA, ALSA, and DLSA models that address the high-dimensionality, sparsity, and redundancy problems in the Wikipedia-based topic space in ELSA by topic feature modelling are described in Section 5.2. Furthermore, LP-ESA that considers both geographical information and personal interests for location-aware news recommendation is presented in Section 5.3. I further introduce LP-DSA, which integrates LP-ESA and DLSA, in Section 5.4. Experimental studies and results are presented and evaluated in Section 5.5.

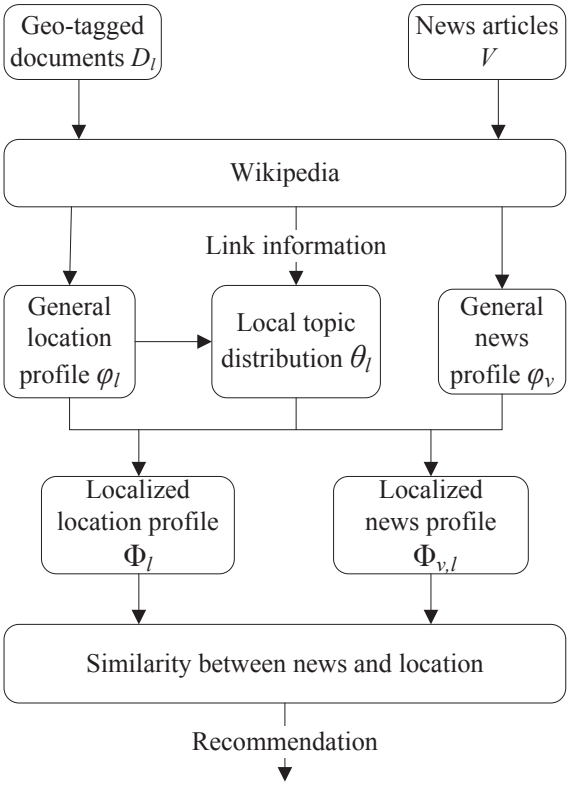


Figure 5.1: The overall process of ELSA

5.1 Explicit Localized Semantic Analysis in News Recommendation

Explicit localized semantic analysis (ELSA) [152] is the state-of-the-art geographical topic-based location-aware news recommendation method which is reported to outperform other geographical topic-based solutions. Due to its close relation to this work, I first present ELSA in this section.

ELSA is an ESA-based [51] solution, where each Wikipedia concept is considered as a potential topic and each location and news is represented as a Wikipedia-based topic vector. Figure 5.1 shows the overall process of ELSA. First of all, ELSA collects

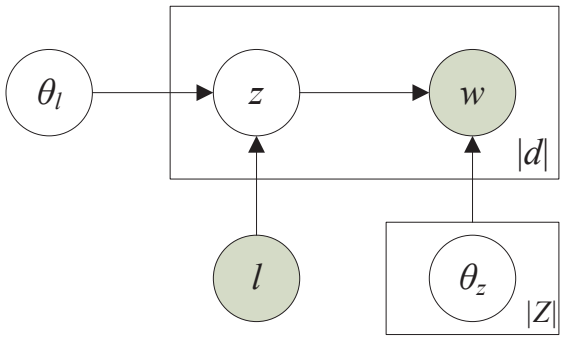


Figure 5.2: Graphical representation of ELSA

for each location a set of documents with the corresponding geo-tags as the description of this location. Then, these geo-tagged documents and the candidate news articles are mapped onto a Wikipedia-based topic space to generate for each location or news a topic vector, which is represented as a probability distribution over topics and called *general location profile* or *general news profile*, respectively. Consequently, the topics related to a location are the ones with non-zero probability values in this location's topic vector. Since these topics are generally dependent on one another, ELSA further uses the link information between the corresponding Wikipedia articles to construct a topic dependency graph and then applies PageRank [120] to estimate a *local topic distribution*. With the help of the local topic distribution, the general location and news profiles are localized to obtain the localized location and news profiles, which are utilized to estimate the similarities between news and locations. Finally, the recommendation is made by offering to the user the news articles with top-k similarity scores to his/her current location. I briefly introduce ELSA in the rest of this section; for the detailed inferences, please refer to [152].

Figure 5.2 shows a graphical representation of ELSA: a document $d \in D_l$ depends on topics $z \in Z$, and the topics are dependent on a location l; also, both d and l are

observed variables, while z, θ_z (i.e., the word distribution of z), and θ_l (i.e., the local topic distribution at l) are all unobserved.

Given U, V, L and Z as the sets of users, news articles, locations and Wikipedia topics, respectively, the general location and news profiles, which are Wikipedia-topic-based vectors, can be formally defined as

$$\varphi_l = \langle p(D_l|z_1, \theta_{z_1}), \ldots, p(D_l|z_{|Z|}, \theta_{z_{|Z|}}) \rangle,$$
$$\varphi_v = \langle p(v|z_1, \theta_{z_1}), \ldots, p(v|z_{|Z|}, \theta_{z_{|Z|}}) \rangle,$$

where $p(D_l|z_i, \theta_{z_i})$ measures how probable D_l is generated from a Wikipedia topic z_i, while $p(v|z_i, \theta_{z_i})$ is the probability of news article v generated from z_i. Then,

$$\begin{aligned} p(D_l|z_i, \theta_{z_i}) &= \prod_{d \in D_l} p(d|z_i, \theta_{z_i}) = \prod_{d \in D_l} \prod_{w \in d} p(w|z_i, \theta_{z_i}) \\ &= \prod_{d \in D_l} \prod_{w \in d} \frac{1 + n_{r_i}(w)}{|W_{r_i}| + \sum\limits_{w_j \in r_i} n_{r_i}(w_j)}, \end{aligned} \tag{5.1}$$

where W_{r_i} is the set of words in r_i, which is a Wikipedia article related to the topic z_i; and $n_{r_i}(w)$ is the frequency of w appearing in r_i. Similarly,

$$p(v|z_i, \theta_{z_i}) = \prod_{w \in v} p(w|z_i, \theta_{z_i}) = \prod_{w \in v} \frac{1 + n_{r_i}(w)}{|W_{r_i}| + \sum_{w \in r_i} n_{r_i}(w)}. \tag{5.2}$$

Moreover, ELSA formally defines a local topic distribution of a location l as

$$\theta_l = \langle p(z_1|l,\theta_l), \ldots, p(z_{|Z_l|}|l,\theta_l)\rangle,$$

where the topics with non-zero $p(D_l|z_i,\theta_{z_i})$ are believed to be the ones actually related to location l and consist the local topic set Z_l. To estimate how important each local topic to location l, ELSA constructs a dependency structure over all topics in Z_l, where the dependency is based on the links between the corresponding articles of the local topics in Wikipedia. Finally, $p(z_i|l,\theta_l)$ is approximated by applying the PageRank algorithm [120] to this structure. Formally,

$$p(z_i|l,\theta_l) \approx PageRank(z_i) = \frac{1-q}{|Z_l|} + \sum_{z_j \in in(z_i,Z_l)} \frac{PageRank(z_j)}{L_{Z_l}(z_j)}, \tag{5.3}$$

where q is a damping factor, $in(z_i, Z_l)$ is the set of topics in Z_l that has links to z_i, and $L_{Z_l}(z_j)$ is the out-degree of z_j.

With the help of local topic distribution, general location profiles, and general news profiles, the localized location profile Φ_l and localized news profile $\Phi_{v,l}$ are defined as

$$\Phi_l = \langle p(D_l, z_1, l|\theta_l, \theta_{z_1}), \ldots, p(D_l, z_{|Z|}, l|\theta_l, \theta_{z_{|Z|}})\rangle, \tag{5.4}$$

$$\Phi_{v,l} = \langle p(v, z_1, l|\theta_l, \theta_{z_1}), \ldots, p(v, z_{|Z|}, l|\theta_l, \theta_{z_{|Z|}})\rangle, \tag{5.5}$$

which can be estimated respectively by the followings:

$$p(D_l, z_i, l|\theta_l, \theta_{z_i}) \quad \propto p(D_l|z_i, \theta_{z_i}) \cdot p(z_i|l, \theta_l), \tag{5.6}$$

$$p(v, z_i, l|\theta_l, \theta_{z_i}) \quad \propto p(v|z_i, \theta_{z_i}) \cdot p(z_i|l, \theta_l). \tag{5.7}$$

Finally, to a given user u, ELSA recommends news articles according to their relevance to u's current location l by simply computing the cosine similarity between Φ_l and $\Phi_{v,l}$.

5.2 Topic Feature Modelling

Although ELSA benefits from using Wikipedia-concept-based topics for semantic enrichment, due to the huge volume of concepts (millions) on Wikipedia, the resulting topic space in ELSA suffers from the problems of high-dimensionality, sparsity, and redundancy, which greatly degrade ELSA's recommendation effectiveness and efficiency.

Therefore, we propose to apply *topic feature modelling* to address these problems and to achieve better performance in location-aware news recommendations. Generally, the process of topic feature modelling takes the general location profile (φ_l), general news profile (φ_v) and local topic distribution (θ_l) in ELSA as input, and exploits either clustering or deep learning techniques to extract dense, abstract, low dimensional, and effective topic features from the Wikipedia-based topic space for the representations of news and locations. The solutions that utilize clustering, autoencoders, and recommendation-oriented deep neural networks for topic feature modelling are presented in the rest of this section.

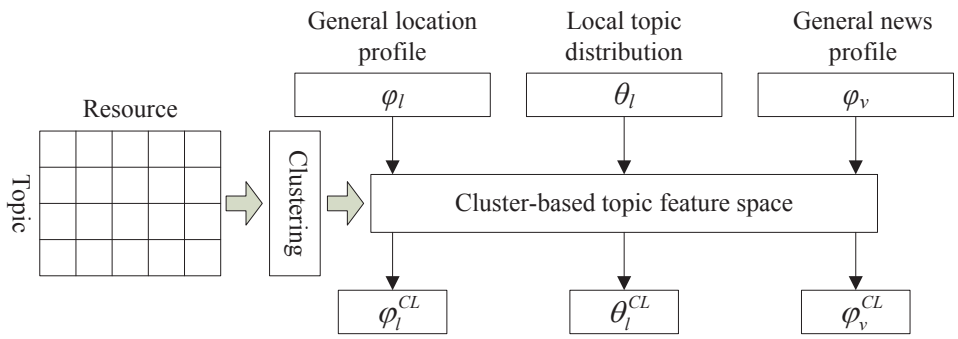


Figure 5.3: Clustering-based topic feature modelling

5.2.1 Clustering-Based Localized Semantic Analysis

Due to its capability in extracting abstract and low dimensional features [148], in this work, we adopt hierarchical clustering as the first solution for the topic feature modelling in ELSA; the resulted geographical topic feature model that integrates clustering with ELSA is called *clustering-based localized semantic analysis (CLSA)*.

As shown in Figure 5.3, the clustering-based topic feature modelling process in CLSA first represents each Wikipedia topic $z \in Z$ as a vector of weights over the set of resources, i.e., the locations and news articles, where the weight on each dimension is measured by the probability of the resource generated from the corresponding topic. Then, hierarchical clustering [148] is applied to assign all topics into a number of clusters based on the distances between their corresponding resource vectors. Finally, with the resulting well-learned clusters, CLSA converts the Wikipedia-based topic space to a cluster-based topic feature space; so the representations of general location profile φ_l, local topic distribution θ_l and general news profile φ_v are converted from Wikipedia-based topic vectors to cluster-based topic feature vectors, which are denoted φ_l^{CL}, θ_l^{CL} and φ_v^{CL}, respectively.

CLSA tackles high-dimensionality, sparsity, and redundancy problems in ELSA

for the following reasons: (i) the cluster-based topic feature space is lower dimensional than the Wikipedia-based topic space in ELSA; (ii) each cluster contains several topics, so sparsity is diminished; and (iii) redundant topics are aggregated to a cluster which reduces redundancy.

As for location-aware recommendation, similarly to ELSA, given φ_l^{CL}, θ_l^{CL}, and φ_v^{CL}, CLSA first obtains the cluster-based localized location and news profiles (denoted Φ_l^{CL} and $\Phi_{v,l}^{CL}$, respectively), which are formally defined as

$$\Phi_l^{CL} = (\varphi_l^{CL})^T \cdot \theta_l^{CL}, \tag{5.8}$$

$$\Phi_{v,l}^{CL} = (\varphi_v^{CL})^T \cdot \theta_l^{CL}. \tag{5.9}$$

Then, given a user at a location l, CLSA generates news recommendations by ranking all news $v \in V$ according to their relevance to l (denoted $R_{l,v}^{CL}$), where the relevance is estimated by the cosine similarity between Φ_l^{CL} and $\Phi_{v,l}^{CL}$. Formally,

$$R_{l,v}^{CL} = Sim(\Phi_l^{CL}, \Phi_{v,l}^{CL}) = \frac{\Phi_l^{CL} \cdot \Phi_{v,l}^{CL}}{\|\Phi_l^{CL}\| \cdot \|\Phi_{v,l}^{CL}\|}. \tag{5.10}$$

5.2.2 Autoencoder-Based Localized Semantic Analysis

Besides clustering, autoencoders are another method to model low dimensional, dense, and abstract representations of raw data [203]. Therefore, in this work, we also employ autoencoders as another solution for the high-dimensionality, sparsity, and re-

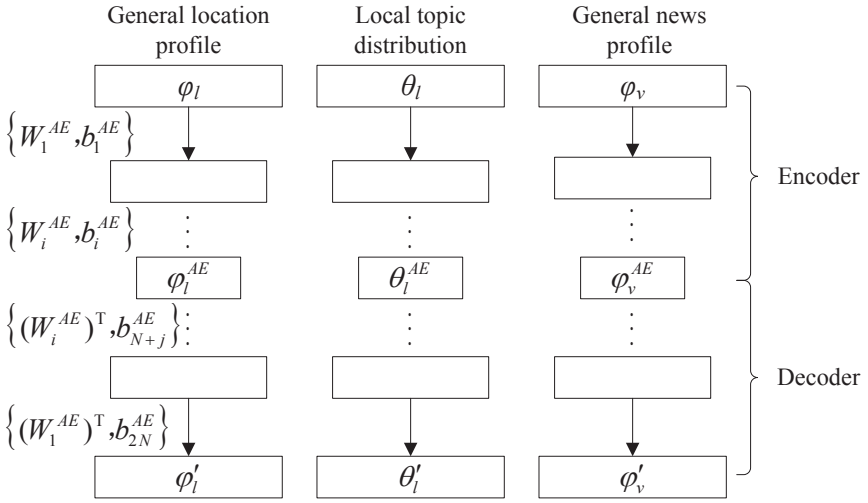


Figure 5.4: Autoencoder-based topic feature modelling

dundancy problems in ELSA; and the resulting geographical topic feature model that integrates autoencoders with ELSA is called *autoencoder-based localized semantic analysis (ALSA)*.

Autoencoders are a kind of neural network consisting of two parts: encoder and decoder. As shown in Figure 5.4, to conduct topic feature modelling, autoencoders first take φ_l, θ_l, and φ_v as inputs, which are passed through multiple hidden layers in encoders. The intermediate outputs $f_i(l)$, $f_i(\theta)$, and $f_i(v)$ of the i-th hidden layers for location, local topics distribution, and news items can be defined formally as

$$f_i(l) = \sigma(W_i^{AE} f_{i-1}(l) + b_i^{AE}), \tag{5.11}$$

$$f_i(\theta) = \sigma(W_i^{AE} f_{i-1}(\theta) + b_i^{AE}), \tag{5.12}$$

$$f_i(v) = \sigma(W_i^{AE} f_{i-1}(v) + b_i^{AE}), \tag{5.13}$$

where $i = 1, \ldots, N$; N is the total number of hidden layers in the encoder (decoder); W_i^{AE} and b_i^{AE} are the weight matrix and bias vector for the i-th hidden layer; $\sigma(\cdot)$ is the sigmoid activation function; and $f_0(l) = \varphi_l$, $f_0(\theta) = \theta_l$, and $f_0(v) = \varphi_v$.

The outputs of the final layers of encoders, $f_N(l)$, $f_N(\theta)$, and $f_N(v)$, are the autoencoder-based topic feature representations for location (denoted φ_l^{AE}), local topic distribution (denoted θ_l^{AE}) and news items (denoted φ_v^{AE}), respectively. Formally

$$\varphi_l^{AE} = f_N(l), \qquad \theta_l^{AE} = f_N(\theta), \qquad \varphi_v^{AE} = f_N(v).$$

Furthermore, the decoders in ALSA take φ_l^{AE}, θ_l^{AE}, and φ_v^{AE} as input and pass them through another N layers. Since tied-weights are used in autoencoders, the weight matrices in the decoder are the transposes of those in the encoder. Formally,

$$f_{N+j}(l) = \sigma((W_{N-(j-1)}^{AE})^T f_{N+(j-1)}(l) + b_{N+j}^{AE}), \tag{5.14}$$

$$f_{N+j}(\theta) = \sigma((W_{N-(j-1)}^{AE})^T f_{N+(j-1)}(\theta) + b_{N+j}^{AE}), \tag{5.15}$$

$$f_{N+j}(v) = \sigma((W_{N-(j-1)}^{AE})^T f_{N+(j-1)}(v) + b_{N+j}^{AE}), \tag{5.16}$$

where $j = 1, \ldots, N$. The outputs of decoders are the reconstructed general location profile, the reconstructed local topic distribution, and the reconstructed general news profile, denoted φ_l', θ_l', and φ_v', respectively. Formally,

$$\varphi'_l = f_{2N}(l), \qquad \theta'_l = f_{2N}(\theta), \qquad \varphi'_v = f_{2N}(v).$$

As for the training of autoencoders, the learning objective of autoencoders in ALSA is to minimize the differences between input and reconstructed data, called *reconstruction errors*. Therefore, the loss function of ALSA is as follows:

$$L^{AE}(\Theta) = \frac{1}{2}\sum_{(l,v)}(\|\varphi'_l - \varphi_l\| + \|\theta'_l - \theta_l\| + \|\varphi'_v - \varphi_v\|), \tag{5.17}$$

where Θ represents the set of parameters $\{W_i^{AE}, b_j^{AE}\}$ $(i = 1, \ldots, N; j = 1, \ldots, 2N)$ in autoencoders.

After training, given the well-modelled autoencoder-based topic feature representations φ_l^{AE}, θ_l^{AE}, and φ_v^{AE}, ALSA first generates the autoencoder-based localized location and news profiles by

$$\Phi_l^{AE} = (\varphi_l^{AE})^T \cdot \theta_l^{AE}, \tag{5.18}$$

$$\Phi_{v,l}^{AE} = (\varphi_v^{AE})^T \cdot \theta_l^{AE}. \tag{5.19}$$

Then, given a user at a location l, ALSA generates location-aware recommendations based on the relevance of all news items $v \in V$ to l, which is computed by the cosine similarity between Φ_l^{AE} and $\Phi_{v,l}^{AE}$. Formally,

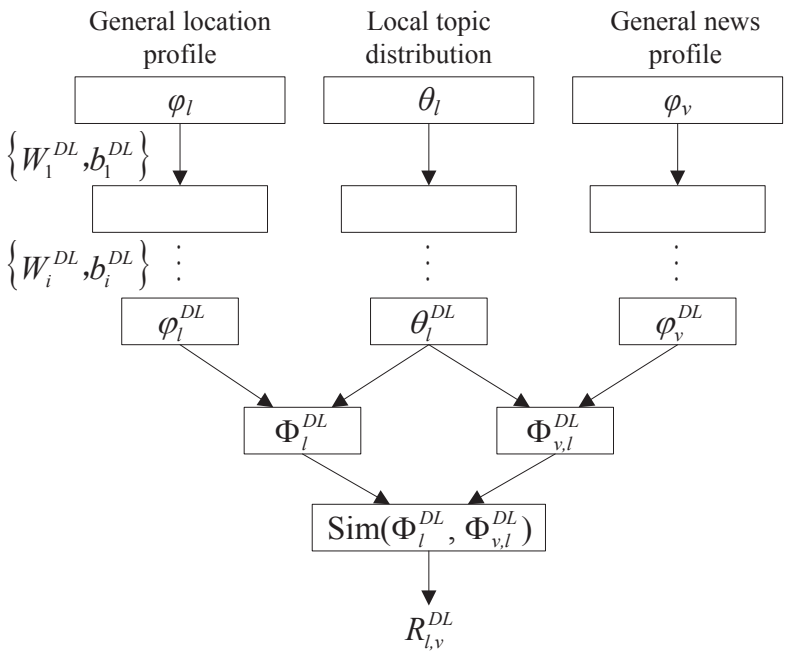


Figure 5.5: Topic feature modelling based on recommendation-oriented deep neural networks

$$R_{l,v}^{AE} = Sim(\Phi_l^{AE}, \Phi_{v,l}^{AE}) = \frac{\Phi_l^{AE} \cdot \Phi_{v,l}^{AE}}{\|\Phi_l^{AE}\| \cdot \|\Phi_{v,l}^{AE}\|}. \tag{5.20}$$

5.2.3 Deep Localized Semantic Analysis

However, CLSA and ALSA suffer from the following drawback: the learning objectives of clustering in CLSA and autoencoders in ALSA are to minimize the within-cluster distances and to minimize the reconstruction errors, respectively, which are not directly correlated to the objective of news recommendation, i.e., distinguishing the users' local target news from the irrelevant ones; so the resulting clustering-based and autoencoder-based topic feature representations of news and locations may not be very

effective in news recommendation.

Therefore, motivated by our findings in Chapter 4, we further propose another geographical topic feature model, called *deep localized semantic analysis (DLSA)*, to address high-dimensionality, sparsity, and redundancy problems in ELSA. DLSA also applies deep neural networks for topic feature modelling; however, instead of using autoencoders, DLSA integrates ELSA with recommendation-oriented deep neural networks as introduced in Chapter 4, which maps the Wikipedia-based topic space to an abstract, dense, and low dimensional topic feature space where the localized similarities between the locations and the users' local target (resp., irrelevant) news are maximized (resp., minimized). Since the deep neural networks in DLSA are trained with a recommendation-oriented learning objective, i.e., to differentiate the users' local target news from the irrelevant ones, the resulting deep topic feature representations of news and locations in DLSA are more effective for location-aware news recommendation than CLSA and ALSA.

As shown in Figure 5.5, similarly to the encoders in ALSA, the three deep neural networks in DLSA also take φ_l, θ_l, and φ_v as inputs and the intermediate outputs h_i of the i-th hidden layers are formally defined as follows:

$$h_i(l) = \tan(W_i^{DL} h_{i-1}(l) + b_i^{DL}), \tag{5.21}$$

$$h_i(\theta) = \tan(W_i^{DL} h_{i-1}(\theta) + b_i^{DL}), \tag{5.22}$$

$$h_i(v) = \tan(W_i^{DL} h_{i-1}(v) + b_i^{DL}), \tag{5.23}$$

where $i = 1, \ldots, N$; $\tan$ is used as the activation function; and $h_0(l) = \varphi_l$, $h_0(\theta) = \theta_l$, $h_0(v) = \varphi_v$. Furthermore, the intermediate outputs in the N-th hidden layers are the

deep topic feature representations for the general location profile (denoted φ_l^{DL}), local topics distribution (denoted θ_l^{DL}) and general news profile (denoted φ_v^{DL}); formally,

$$\varphi_l^{DL} = h_N(l), \qquad \theta_l^{DL} = h_N(\theta), \qquad \varphi_v^{DL} = h_N(v).$$

Given φ_l^{DL}, θ_l^{DL}, and φ_v^{DL}, the deep localized location and news profiles are formally defined as

$$\Phi_l^{DL} = (\varphi_l^{DL})^T \cdot \theta_l^{DL}, \tag{5.24}$$

$$\Phi_{v,l}^{DL} = (\varphi_v^{DL})^T \cdot \theta_l^{DL}. \tag{5.25}$$

Then, for a user at a location l, the similarity between l and a news article v is measured using the cosine similarity between their deep localized profiles at l, i.e., Φ_l^{DL} and $\Phi_{v,l}^{DL}$; formally,

$$Sim(\Phi_l^{DL}, \Phi_{v,l}^{DL}) = \frac{\Phi_l^{DL} \cdot \Phi_{v,l}^{DL}}{\|\Phi_l^{DL}\| \cdot \|\Phi_{v,l}^{DL}\|}. \tag{5.26}$$

Differently from CLSA and ALSA, instead of using the cosine similarity directly, the relevance scores of news items v to given users at locations l are measured by applying the softmax function on the resulting similarities of all news at l, which are then used to generate location-aware recommendation lists. Formally,

$$R_{l,v}^{DL} = \frac{e^{Sim(\Phi_l^{DL}, \Phi_{v,l}^{DL})}}{\sum\limits_{v' \in V} e^{Sim(\Phi_l^{DL}, \Phi_{v',l}^{DL})}}. \tag{5.27}$$

Intuitively, to achieve good location-aware news recommendations, the local target news should have higher relevance scores than irrelevant ones. We thus conduct the model training in DLSA with a recommendation-oriented objective to maximize the relevance scores of local target news; equivalently, it is to maximize the localized similarities between locations and their local target news and minimize those with irrelevant ones. Formally, it is equivalent to minimize the following loss function:

$$\begin{aligned} L^{DL}(\Theta) &= -\sum_{(l,v^*)} \log(R_{l,v^*}^{DL}) \\ &= -\sum_{(l,v^*)} \left[\log(e^{Sim(\Phi_l^{DL}, \Phi_{v^*,l}^{DL})}) - \log(\sum_{v' \in V} e^{Sim(\Phi_l^{DL}, \Phi_{v',l}^{DL})}) \right], \end{aligned} \tag{5.28}$$

where Θ is the set of parameters $\{W_i^{DL}, b_i^{DL}\}$ $(i = 1, \ldots, N)$ in DLSA; the tuple (l, v^*) is a training sample, indicating that v^* is a local target news to the user at location l.

As for the training of DLSA (resp., ALSA), we first initialize the weight matrices W_i^{DL} (resp., W_i^{AE}) using the random normal distribution and initialize the bias vectors b_i^{DL} (resp., b_i^{AE}) to be zero vectors; the model is then trained via mini-batch gradient descent [29], which is a gradient-based optimization algorithm; finally, the training stops when the model converges or reaches the maximum training iterations.

In summary, applying deep neural networks for topic feature modelling in DLSA and ALSA is capable to overcome high-dimensionality, sparsity, and redundancy prob-

lems in ELSA, because of the following reasons: (i) The number of nodes in the hidden layer is much smaller than that in the input layer, so the dimensionality of the resulting deep (or autoencoder-based) topic feature space is much lower than that of the Wikipedia-based topic space in ELSA; and (ii) deep neural networks in DLSA and ALSA extract more abstract and denser features layer-by-layer, so sparsity and redundancy problems are addressed.

5.3 Location-Aware Personalized News Recommendation With Explicit Semantic Analysis

On the other hand, ELSA has the other shortcoming as follows: since ELSA and most of the location-aware news recommendation methods consider solely the user's location for news recommendation; the recommended news for different users at the same location will remain the same. However, in practice, a users' real-time news preferences are strongly correlated not only to their current locations but also to their long-term personal interests, so both should be considered to achieve satisfactory personalized news recommendations.

Therefore, we extend ELSA to propose a novel news recommendation model, called *location-aware personalized news recommendation with explicit semantic analysis* (LP-ESA), which takes into account both the user's current location and also his/her personal interests. Consequently, differently from ELSA, in LP-ESA, different users at a same location will have different and personalized news recommendations. LP-ESA is similar to ELSA but replaces the location profile with a user profile for personalization, and location is only responsible for localization.

Generally, LP-ESA is a personalized extension of ELSA: it replaces the location

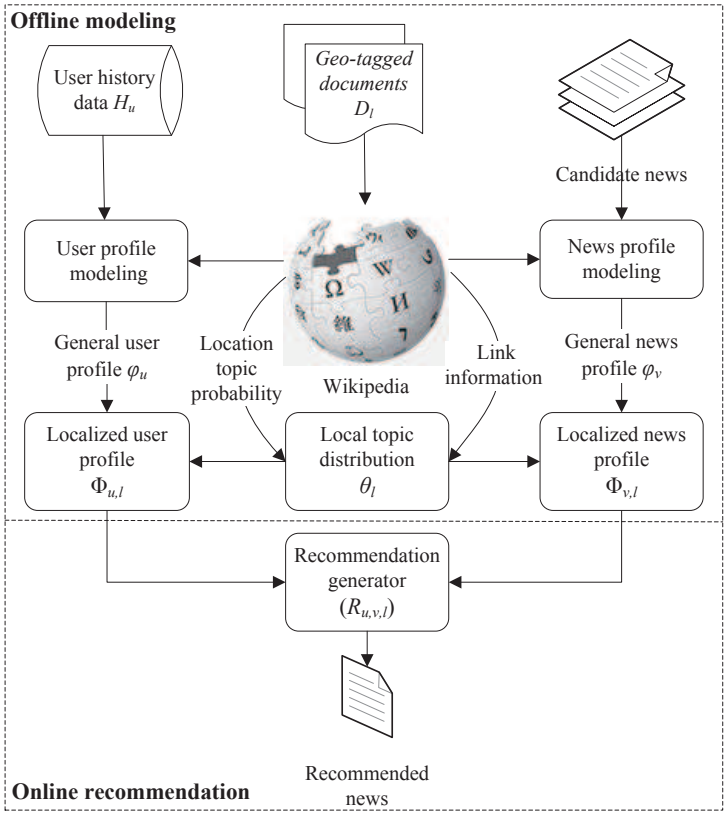


Figure 5.6: The overall process of LP-ESA

profile with a user profile for personalization, and the location information is used for localization via local topic distribution. The overall process of LP-ESA is depicted in Figure 5.6, which generally consists of two steps: offline modeling and online recommendation. Offline modeling aims to construct the localized profiles for users and news, which are then used to conduct location-aware personalized news recommendation in the online recommendation step.

Specifically, in the offline modeling step, given a user u , LP-ESA first projects the user's history data H_u and each news article v in the news set V onto a Wikipedia topic space to construct a *general user profile* φ_u for the given user u and a *general news profile* φ_v for each news v, respectively. However, both profiles are insensitive to

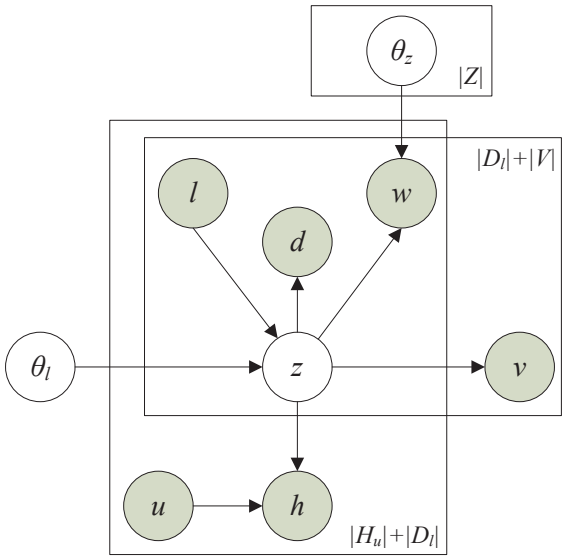


Figure 5.7: Graphical representation of LP-ESA

the change of the user's geographical context. To achieve location-aware personalized news recommendation, given a location l, LP-ESA then models a *localized user (resp., news) profile* $\Phi_{u,l}$ (resp., $\Phi_{v,l}$) for each user u (resp. news v) at location l, according to both the general user (resp., news) profile φ_u (resp., φ_v) and the local topic distribution of l (i.e., θ_l), which is estimated using the same process as in ELSA. Finally, both localized user and news profiles are utilized in the online recommendation phrase to generate for the given user u a ranked list of news according to the relevancy score $R_{u,v,l}$, which is computed by the cosine similarity of $\Phi_{u,l}$ and $\Phi_{v,l}$. Formally,

$$R_{u,v,l} = Sim(\Phi_{u,l}, \Phi_{v,l}) = \frac{\Phi_{u,l} \cdot \Phi_{v,l}}{\|\Phi_{u,l}\| \cdot \|\Phi_{v,l}\|}. \tag{5.29}$$

The graphical representation of LP-ESA is shown in Figure 5.7, where a geo-tagged

document $d \in D_l$, a history data record $h \in H_u$, a news article $v \in V$, and a word w are all dependent on topic $z \in Z$, while the topics depend on a location l. In addition, l, u, d, h, and v are observed variables, while z, θ_z, and θ_l are the unobserved ones.

5.3.1 General User Profile

As the key of personalized news recommendation, for a given user, a user profile is used to describe this user's personal interests on news, which can be modeled using his/her history data on the Web (e.g., query history, browsing history, or tweeting history). Therefore, LP-ESA first collects a set of tweeting history data H_u for each given user u and then projects H_u onto the Wikipedia topic space to model u's general user profile (denoted φ_u); formally,

$$\varphi_u = \langle p(H_u|z_1, \theta_{z_1}), \ldots, p(H_u|z_i, \theta_{z_i}), \ldots, p(H_u|z_{|Z|}, \theta_{z_{|Z|}}) \rangle, \tag{5.30}$$

where $p(H_u|z_i, \theta_{z_i})$ is the probability of the topic z_i in the tweeting history data H_u of u. Then, given $h \in H_u$ as a tweeting record,

$$p(H_u|z_i, \theta_{z_i}) = \prod_{h \in H_u} p(h|z_i, \theta_{z_i});$$

then, we have:

$$\log p(H_u|z_i, \theta_{z_i}) = \sum_{h \in H_u} \log p(h|z_i, \theta_{z_i}), \tag{5.31}$$

where $p(h|z_i, \theta_{z_i})$ is the probability of a topic z_i in the tweeting record h.

Differently from ELSA, where the probability of a topic in a geo-tagged document $p(D_l|z_i, \theta_{z_i})$ is estimated solely based on the possibility that the words in the geo-tagged document are generated from this topic, i.e., $p(w|z_i, \theta_{z_i})$ (as shown in Formula (5.1)), LP-ESA considers both the probability of generating a tweeting record h from each word in h (i.e., $p(h|w)$) and that of the word w generated from each topic z_i (i.e., $p(w|z_i, \theta_{z_i})$) to estimate the probability of a topic z_i in the tweeting record h (i.e., $p(h|z_i, \theta_{z_i})$). Consequently, the word with higher probability in h contributes more to the topic probability estimation in LP-ESA; while all words of the document are equally important in ELSA. Therefore, assuming all words are independent, there is the following derivation:

$$\log p(h|z_i, \theta_{z_i}) = \sum_{w \in h} \log p(h|w) p(w|z_i, \theta_{z_i}), \tag{5.32}$$

where both $p(h|w)$ and $p(w|z_i, \theta_{z_i})$ are estimated using a variant of TF-IDF [161] in LP-ESA; formally,

$$p(h|w) = \frac{(1 + \log n_h(w)) * \log \frac{|Z|}{n_Z(w)}}{\sqrt{\sum_{w_j \in h} [(1 + \log n_h(w_j)) * \log \frac{|Z|}{n_Z(w_j)}]^2}}, \tag{5.33}$$

$$p(w|z_i, \theta_{z_i}) = \frac{(1 + \log(n_{r_i}(w) + 1)) * \log \frac{|Z|}{n_Z(w)}}{\sqrt{\sum_{w_j \in r_i} [(1 + \log(n_{r_i}(w_j) + 1)) * \log \frac{|Z|}{n_Z(w_j)}]^2} + \log |Z|}, \tag{5.34}$$

where $n_h(w)$ and $n_{r_i}(w)$ denote the frequency of w appearing in h and r_i (a Wikipedia article with regard to topic z_i), respectively; and $n_Z(w)$ is the number of topics in Z whose related Wikipedia articles contain w.

5.3.2 Localized User Profile

As the users' news preferences usually evolve with the change of their locations, to achieve a location-aware personalized news recommendation, for a given user u at a location l, LP-ESA further constructs a localized user profile based on u's general user profile φ_u and the local topic distribution θ_l of l, where θ_l is estimated using the same process as in ELSA (see Section 5.1). Formally, the localized user profile of user u at location l is defined as

$$\Phi_{u,l} = \langle p(H_u, z_1, l|\theta_l, \theta_{z_1}), \ldots, p(H_u, z_i, l|\theta_l, \theta_{z_i}), \ldots, p(H_u, z_{|Z|}, l|\theta_l, \theta_{z_{|Z|}}) \rangle,$$

where $p(H_u, z_i, l|\theta_l, \theta_{z_i})$ measures the probability of u's preferences w.r.t. topic z_i at location l. Since z depends on l and θ_l, while l is independent of θ_l and θ_z (as shown in Figure 5.7), by assuming $p(l)$ to be uniform, we then estimate $p(H_u, z_i, l|\theta_l, \theta_{z_i})$

according to the following deduction:

$$\begin{aligned} p(H_u, z_i, l|\theta_l, \theta_{z_i}) &= p(H_u|z_i, \theta_{z_i}) \cdot p(z_i|l, \theta_l) \cdot p(l) \\ &\propto p(H_u|z_i, \theta_{z_i}) \cdot p(z_i|l, \theta_l); \end{aligned} \tag{5.35}$$

then, we have

$$\log p(H_u, z_i, l|\theta_l, \theta_{z_i}) \propto \sum_{h \in H_u} \log p(h|z_i, \theta_{z_i}) + \log p(z_i|l, \theta_l), \tag{5.36}$$

where $p(z_i|l, \theta_l)$ is the local topic probability in the local topic distribution θ_l, and $p(H_u|z_i, \theta_{z_i})$ is the topic probability in the general user profile φ_u.

5.3.3 General News Profile

Similarly, LP-ESA also models a general news profile (φ_v) for each news article v; formally,

$$\varphi_v = \langle p(v|z_1, \theta_{z_1}), \ldots, p(v|z_i, \theta_{z_i}), \ldots, p(v|z_{|Z|}, \theta_{z_{|Z|}}) \rangle .$$

Although the definition of the general news profile looks the same as the definition of the general news profile in ELSA (see Formula (5.2)), the estimation of $p(v|z_i, \theta_{z_i})$ here is actually different from that in ELSA. Specifically, LP-ESA utilizes both the

probability of obtaining a news article v given each word w in v (i.e., $p(v|w)$) and that of each word w in v generated from each topic z_i (i.e., $p(w|z_i, \theta_{z_i})$) to estimate $p(v|z_i, \theta_{z_i})$; while ELSA merely considers the latter factor. Therefore, Similarly to Formula (5.32), $p(v|z_i, \theta_{z_i})$ is estimated as follows:

$$\log p(v|z_i, \theta_{z_i}) = \sum_{w \in v} \log p(v|w) p(w|z_i, \theta_{z_i}),$$

where $p(w|z_i, \theta_{z_i})$ can be estimated by Formula (5.34), and $p(v|w)$ can be estimated based on a variant of TF-IDF as follows,

$$p(v|w) = \frac{(1 + \log n_v(w)) * \log \frac{|Z|}{n_Z(w)}}{\sqrt{\sum_{w_j \in v} [(1 + \log n_v(w_j)) * \log \frac{|Z|}{n_Z(w_j)}]^2}}.$$

5.3.4 Localized News Profile

Based on the general news profile φ_v and the local topic distribution θ_l at a location l, the localized news profile of a news article v at l is defined as follows:

$$\Phi_{v,l} = \langle p(v, z_1, l|\theta_l, \theta_{z_1}), \ldots, p(v, z_i, l|\theta_l, \theta_{z_i}), \ldots, p(v, z_{|Z|}, l|\theta_l, \theta_{z_{|Z|}}) \rangle,$$

where $p(v, z_i, l|\theta_l, \theta_{z_i})$ is the probability of topic z_i in news v at l. Similarly to Formula (5.36),

$$p(v, z_i, l|\theta_l, \theta_{z_i}) = p(v|z_i, \theta_{z_i}) \cdot p(z_i|l, \theta_l) \cdot p(l)$$
$$\propto p(v|z_i, \theta_{z_i}) \cdot p(z_i|l, \theta_l), \tag{5.37}$$

where $p(v|z_i, \theta_{z_i})$ is the topic probability in the general news profile φ_v, and $p(z_i|l, \theta_l)$ is the local topic probability in θ_l.

5.4 Location-Aware Personalized News Recommendation with Deep Semantic Analysis

Nevertheless, since LP-ESA is also an ESA-based method, similarly to ELSA, it also suffers from high-dimensionality, sparsity, and redundancy problems. To solve these problems and further improve the performance of LP-ESA, we thus propose a hybrid news recommendation model called *location-aware personalized news recommendation with deep semantic analysis* (LP-DSA), which integrates the intuition of DLSA with LP-ESA to utilizes deep neural networks to map the Wikipedia-based topic space in LP-ESA to an abstract, dense, and low dimensional feature space, where the localized similarities between the users and their target news are maximized, and those with irrelevant news are minimized. LP-DSA is a deep learning extension of LP-ESA as well as a personalized extension of DLSA; so it has the advantages of both models.

Figure 5.8 depicts the overall process of LP-DSA. First of all, LP-DSA takes the general user profile φ_u, local topic distribution θ_l, and general news profile φ_v (generated in LP-ESA) as the respective inputs of three deep neural networks with shared

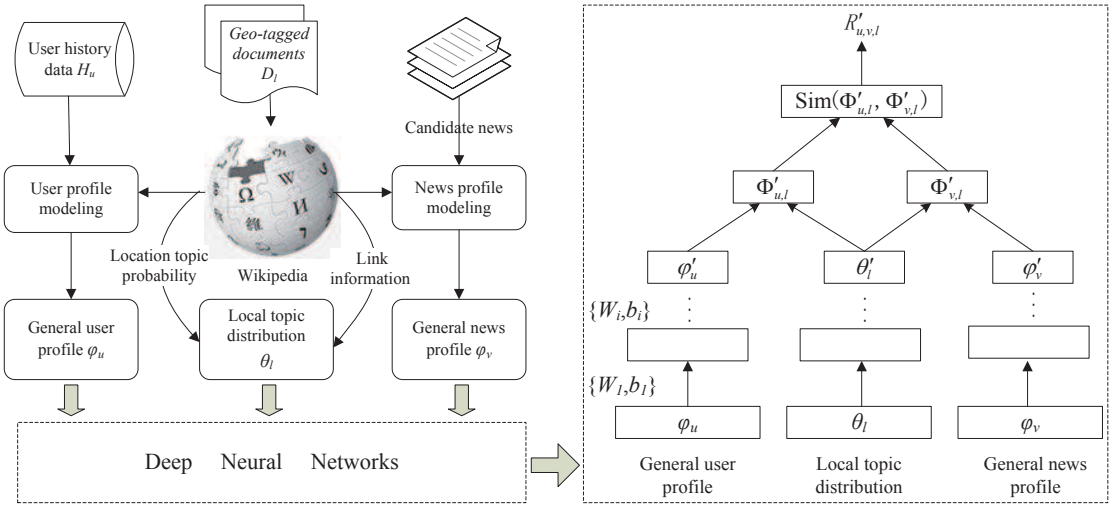


Figure 5.8: The overall process of location-aware personalized news recommendation with deep semantic analysis

parameters. Then these inputs are then passed through multiple hidden layers and projected onto an abstract, dense, and low dimensional feature space on the final hidden layer, resulting in three abstract general feature representations for the user, location, and news item, which are called *abstract general user profile* (denoted φ'_u), *abstract local topic distribution* (denoted θ'_l), and *abstract general news profiles* (denoted φ'_v), respectively. Then, LP-DSA models an *abstract localized user profile* (denoted $\Phi'_{u,l}$) using φ'_u and θ'_l and an *abstract localized news profile* (denoted $\Phi'_{v,l}$) using φ'_v and θ'_l. Finally, for a given user, LP-DSA recommends to this user the news with top-k highest relevance scores, which are computed by applying the softmax function on cosine similarities of the corresponding abstract localized profiles of the user and news. To learn the parameters W_i and b_i in these deep neural networks, the model is trained using an objective to maximize the deep-semantic similarities between the users and their target news, while minimizing those with irrelevant ones.

Formally, given a weight matrix W_1 and a bias vector b_1, LP-DSA takes the general

user profile φ_u, local topic distribution θ_l, and general news profile φ_v as the input and generates the intermediate outputs f_1 of the first hidden layer according to the following formulas:

$$f_1(u) = \tanh(W_1\varphi_u + b_1), \tag{5.38}$$

$$f_1(l) = \tanh(W_1\theta_l + b_1), \tag{5.39}$$

$$f_1(v) = \tanh(W_1\varphi_v + b_1), \tag{5.40}$$

where $\tanh$ is adopted as the activation function. Similarly, the intermediate output of the i-th hidden layer ($i = 2, \ldots, N$) are computed by

$$f_i(u) = \tanh(W_i f_{i-1}(u) + b_i), \tag{5.41}$$

$$f_i(l) = \tanh(W_i f_{i-1}(l) + b_i), \tag{5.42}$$

$$f_i(v) = \tanh(W_i f_{i-1}(v) + b_i), \tag{5.43}$$

where W_i and b_i are the weight matrix and bias vector for the i-th hidden layer, and N is the total number of hidden layers. Consequently, the output of the N-th hidden layer are the abstract general user profile φ'_u, the abstract local topic distribution θ'_l, and the abstract general news profile φ'_v. Formally,

$$\varphi'_u = f_N(u), \quad \theta'_l = f_N(l), \quad \varphi'_v = f_N(v). \tag{5.44}$$

Furthermore, since φ'_u, θ'_l, φ'_v share the same feature space, according to Formulas (5.36) and (5.37), the abstract localized user and news profiles can be formally defined as

$$\Phi'_{u,l} = \varphi'^{T}_{u} \cdot \theta'_l, \tag{5.45}$$

$$\Phi'_{v,l} = \varphi'^{T}_{v} \cdot \theta'_l. \tag{5.46}$$

Then, the similarity between a user and a news article at location l is measured using the cosine similarity between their abstract localized profiles at l; formally,

$$Sim(\Phi'_{u,l}, \Phi'_{v,l}) = \frac{\Phi'_{u,l} \cdot \Phi'_{v,l}}{\|\Phi'_{u,l}\| \cdot \|\Phi'_{v,l}\|}. \tag{5.47}$$

Finally, LP-DSA recommends news articles based on their relevance scores to a given user u at location l, where the relevance scores are computed by applying the softmax function on the resulting similarities; formally,

$$R'_{u,v,l} = \frac{e^{Sim(\Phi'_{u,l}, \Phi'_{v,l})}}{\sum\limits_{v' \in V} e^{Sim(\Phi'_{u,l}, \Phi'_{v',l})}}, \tag{5.48}$$

Therefore, to achieve a good personalized news recommendation, the target news should have higher relevance scores than others. We thus conduct the model training

with an objective to maximize the relevance scores of target news; equivalently, it is to maximize the localized similarities between users and their target news and minimize those with irrelevant ones. Formally, it is equivalent to minimize the following loss function:

$$\begin{aligned} Loss(\Theta) &= -\sum_{(u,v^*,l)} \log(R'_{u,v^*,l}) \\ &= -\sum_{(u,v^*,l)} \left[\log(e^{Sim(\Phi'_{u,l},\Phi'_{v^*,l})}) - \log(\sum_{v' \in V} e^{Sim(\Phi'_{u,l},\Phi'_{v',l})}) \right], \end{aligned} \tag{5.49}$$

where Θ represents the set of parameters $\{W_i, b_i\}$ $(i = 1, \ldots, N)$ in the neural networks; tuple (u, v^*, l) is a training sample, which represents that a user u has a target news v^* at the location l and is generated from the user's tweeting history records in the training dataset.

According to the above loss function, ideally, all news in V should be used as candidate news in the model training. However, due to the large number of candidate news, this will result in a very expensive training cost. Therefore, in practice, to train the model more efficiently, for each training sample (u, v^*, l), we approximate V by including the target news v^* and only K randomly selected irrelevant news v^-. Our pilot study in Chapter 4 shows that this approximation has few damage to the training effectiveness.

Table 5.1: Statistics information of dataset

Tweets	Users	News	Locations	Samples
2,316,204	1,619	63,485	2,366	98,321

Table 5.2: Details of training set and test (sub)sets

	Users	News	Locations	Samples
Training set	1,555	53,036	2,295	78,946
Test set	909	15,853	743	19,375
Old City test subset	825	12,518	325	14,883
New City test subset	788	3,822	662	4,492

5.5 Experiments

In this section, I report on experimental evaluations of the proposed methods; I first describe the experimental settings in Section 5.5.1; then the effect of applying topic feature modelling and the performance evaluation of DLSA, ALSA, and CLSA is in Section 5.5.2; in addition, the effect of considering users' personal interests and the performance evaluation of LP-ESA is in Section 5.5.3; finally, I evaluate the performance of the hybrid model, LP-DSA, in Section 5.5.4

5.5.1 Experimental Setting

Dataset and Preprocessing

This experimental study is based on a publicly available Twitter dataset [7], which consists of $2,316,204$ tweets posted by $1,619$ users ($|U| = 1,619$). About half (over one million) of these tweets explicitly contain URLs to the news articles; by using these URLs to download the corresponding news articles, we get $63,485$ news articles, which

are used as the candidate news articles for recommendation, i.e., $|V| = 63,485$. Then, we apply a Web service tool[1] to extract city names from the news articles, resulting in $2,366$ locations ($|L| = 2,366$). Finally, we consider these city names as geo-tags and use the titles and keywords of the news articles, from which the city names are extracted, as the descriptions of these locations, i.e., geo-tagged documents D_l. The statistic information of the resulting dataset is summarized in Table 5.1.

To model the users' news preferences, if a user u posts a tweet which has an URL to a news article v^* containing a city name (location) l, u is believed to be interested in v^* at l, from which a sample (u, v^*, l) is generated, indicating v^* is a target news of u at l. Consequently, a total of $98,321$ samples are obtained from the dataset; we randomly select 80% the samples as the training set and the remaining 20% as the test set. We further divide the test set into two subsets: Old City test subset and New City test subset, to evaluate the methods' news recommendations performance under different geographical context. This is because, in practice, it is generally more difficult to achieve satisfactory news recommendation when users are at new locations where they have never visited and do not have related local history data. Specifically, for each sample (u, v^*, l) in the test set, if the training set also contains some samples related to the same user u at the same location l, l is seen as a city (location) where u has already visited before so (u, v^*, l) is added to Old City test subset; otherwise, l is a new city to user u and the sample (u, v^*, l) is added to New City test subset. The details of training set and test (sub)sets are summarized in Table 5.2.

Furthermore, a news recommendation system always aims to recommend to users news that they have never read before. Therefore, to be in line with this requirement, we remove the tweets where test samples are extracted and use only the rest of the

[1]OpenCalais at https://opencalais.com/

tweeting records in the dataset as the users' history data (i.e., H_u) to construct user profiles. Finally, a Wikipedia snapshot of August $11, 2014$ is used for the process of explicit semantic analysis [51], resulting in $1,301,900$ concepts with $1,618,970$ distinct terms. To cut down the calculation and memory cost, we select $8,000$ most frequent topics/concepts among all the resulted topic representations of the users, locations, and news items as the shared Wikipedia topic space.

Parameter Settings and Evaluation Metrics

All methods are implemented using Python and Theano (a Python library for machine learning) and run on a GPU server with a NVIDIA Tesla K20 and 8GB graphics card memory. ELSA is implemented based on [152]; the parameters of the proposed models are set empirically as follows:

- The damping factor q is set to 0.85.
- The number of clusters in CLSA is empirically set to 1024.
- The number of hidden layers in DLSA and LP-DSA and in the encoder of ALSA is $N = 3$.
- The number of neurons in the first, second, and third hidden layers is 1024, 512 and 256, respectively.
- The number of news that is randomly selected to approximate V for each training sample is $K = 127$;
- The learning rate in model training is 0.0001.
- ALSA adds another two hidden layers for the decoder, and the number of neurons in the fourth and fifth layers is 512 and 1024, respectively.

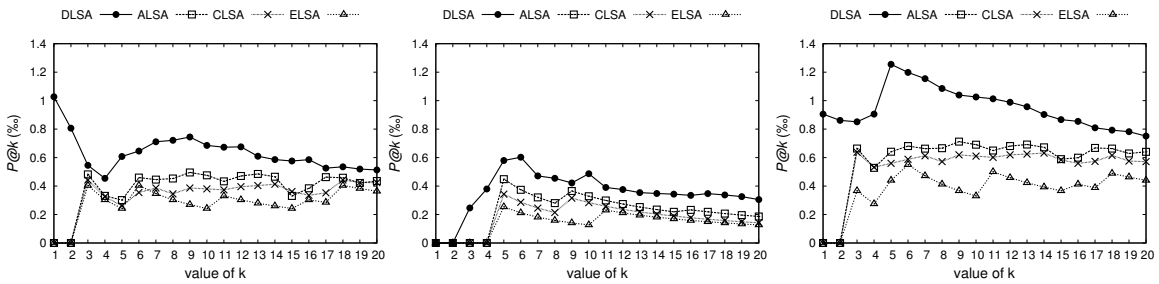
The commonly used *precision at k ($P@k$)*, *recall at k ($R@k$)*, *F1-score at k ($F@k$)*, and *mean reciprocal rank (MRR)* as defined in Section 4.3 of Chapter 4 are adopted for the evaluation of the location-aware news recommendation systems.

5.5.2 Effect of Applying Topic Feature Modelling

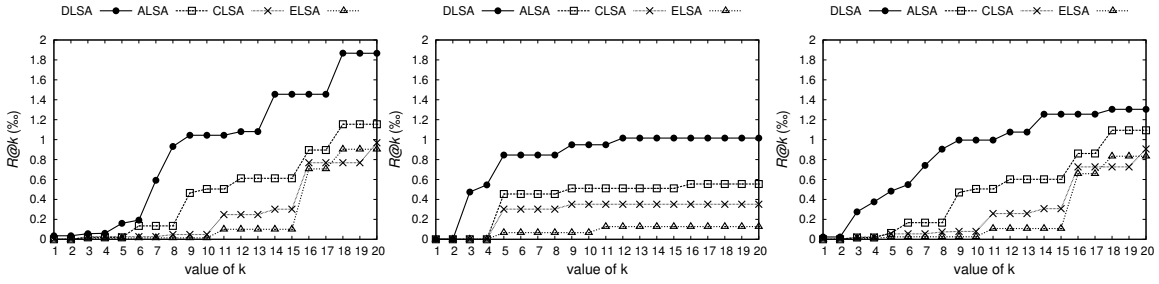
We first evaluate the effect of applying topic feature modelling to address high-dimensionality, sparsity, and redundancy problems in the Wikipedia-based topic space in ELSA. Figure 5.9(a)-(i) depict the news recommendation performances of DLSA, ALSA, CLSA, and ELSA on three test (sub)sets in terms of precision at k ($P@k$), recall at k ($R@k$), and F1-score at k ($F@k$), where the value of k varies from 1 to 20; while the results of mean reciprocal rank (MRR) of DLSA, ALSA, CLSA and ELSA on three test (sub)sets are also shown in Figure 5.9(j)-(l).

Generally, in Figure 5.9, the proposed three geographical topic feature models, DLSA, ALSA and CLSA, all greatly outperform the Wikipedia-based topic model, ELSA, in location-aware news recommendations in terms of all evaluation metrics on all three test (sub)sets. This finding demonstrates that applying clustering or deep learning techniques for topic feature modelling can address high-dimensionality, sparsity and redundancy problems in ELSA and greatly enhances the performance of location-aware new recommendation.

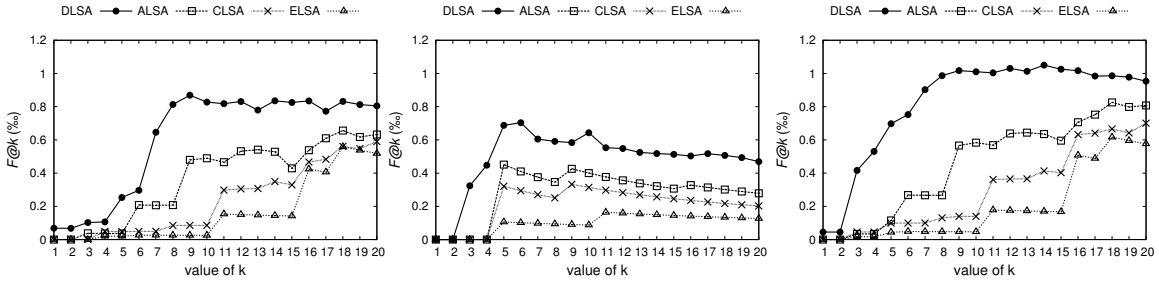
Figures 5.9(a)-(c) exhibit the precision at k ($P@k$) of all the four models on three test (sub)sets. Specifically, Figure 5.9(a) presents the results on Old City test subset; it is shown that DLSA achieves the best performance in $P@k$ for all k values while ALSA and CLSA also outperforms ELSA constantly. The values of $P@k$ for all methods are relatively stable with the change of k. Figure 5.9(b) and (c) depict the $P@k$ results of DLSA, ALSA, CLSA, and ELSA on the New City test subset and the whole test



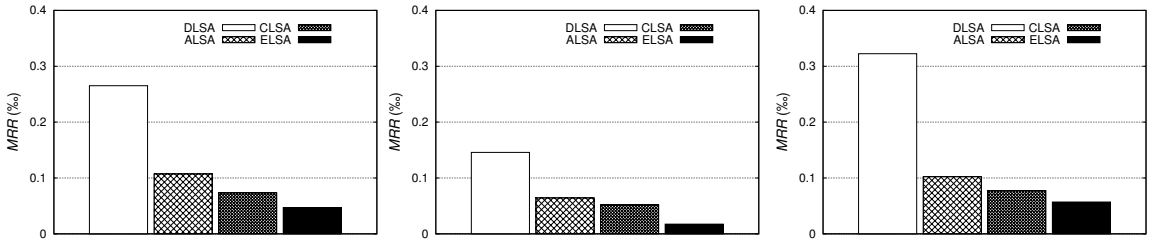
(a) $P@k$ on Old City test subset (b) $P@k$ on New City test subset (c) $P@k$ on the whole test set



(d) $R@k$ on Old City test subset (e) $R@k$ on New City test subset (f) $R@k$ on the whole test subset



(g) $F@k$ on Old City test subset (h) $F@k$ on New City test subset (i) $F@k$ on the whole test set



(j) MRR on Old City test subset (k) MRR on New City test subset (l) MRR on the whole test set

Figure 5.9: The news recommendation effectiveness of DLSA, ALSA, CLSA, and ELSA on three test (sub)sets in terms of precision at k ($P@k$), recall at k ($R@k$), F1-score at k ($F@k$), and mean reciprocal rank (MRR), with scales on y-axis in ‰

set, respectively, where the general tendency and the comparison results between these four models are similar to those in Figure 5.9(a) with a peak value for DLSA at $k = 6$.

Figures 5.9(d)-(f) show the models' recall at k ($R@k$) on the Old City test subset, the New City test subset and the whole test set, respectively. Similarly, DLSA is also very superior to all other methods in $R@k$ on the three test (sub)sets, and ALSA and CLSA holds the second and third places again. However, differently from $P@k$, the four methods all witness increases in $R@k$ with the rise of k's value on the three (sub)sets and the increases of DLSA, ALSA, and CLSA are generally more significant than those of ELSA. Figures 5.9(g)-(i) are the results of the F1-score at k ($F@k$) on the three test (sub)sets. Since $F@k$ is an integrated metric computed by $R@k$ and $P@k$, we have similar observations that reveal the superiority of DLSA.

The results of the mean reciprocal rank (MRR) on various test (sub)sets are shown in Figures 5.9(j)-(l), where DLSA, ALSA, and CLSA still dramatically outperform the baseline, ELSA. This shows that the recommendation lists generated by DLSA, ALSA, and CLSA have much better ordering than ELSA, which is an important improvement for personalized new recommendation and shows the essential of applying topic feature modelling in ELSA, because finding the target news becomes much easier for users using DLSA, ALSA, and CLSA.

Evaluation and Discussion

Furthermore, to evaluate the general performances of these models, in Table 5.3, we further compute and record the average $P@k$, $R@k$, and $F@k$ of DLSA, ALSA, CLSA and ELSA on three test (sub)sets when $1 \leqslant k \leqslant 20$; while the numerical values of MRR of these models are also recorded in Table 5.3.

Table 5.3 shows that with the help of the recommendation-oriented deep neural

Table 5.3: Average $P@k$, $R@k$, $F@k$, and MRR of DLSA, ALSA, CLSA and ELSA on three test (sub)sets when $1 \leqslant k \leqslant 20$ (in ‰)

	Old City test subset			
	P@k	R@k	F@k	MRR
ELSA	0.2834	0.2361	0.1687	0.0470
CLSA	0.3404	0.2834	0.2345	0.0736
ALSA	0.3896	0.4820	0.3627	0.1073
DLSA	**0.6374**	**0.9389**	**0.6070**	**0.2652**
	New City test subset			
	P@k	R@k	F@k	MRR
ELSA	0.1385	0.0836	0.1009	0.0174
CLSA	0.1793	0.2717	0.2112	0.0520
ALSA	0.2218	0.4085	0.2811	0.0643
DLSA	**0.3551**	**0.8193**	**0.4867**	**0.1458**
	The whole test set			
	P@k	R@k	F@k	MRR
ELSA	0.3773	0.2263	0.1981	0.0569
CLSA	0.5343	0.2822	0.2995	0.0774
ALSA	0.5863	0.4745	0.4553	0.1023
DLSA	**0.9500**	**0.8723**	**0.8223**	**0.3224**

networks, DLSA achieves much better recommendation performance than ALSA and CLSA. For example, on the whole test set, the average precision of DLSA is roughly double of those of ALSA and CLSA; the average recall and F1-score of DLSA are both roughly double of those of ALSA and about three times of those of CLSA; and the MRR of DLSA is roughly three times of that of ALSA and about four times of that of CLSA. The superior performance of DLSA is mainly because DLSA uses a recommendation-oriented learning objective, which is directly correlated with distinguishing the user's local target news items from the irrelevant ones; so, the resulting deep topic features for the representations of news and location profiles in DLSA are much more effective for news recommendations than those generated by clustering and

autoencoders, whose learning objectives are not directly related to recommendations. These results are consistent with the experimental results of tag-aware personalized recommendation in Chapter 4 where recommendation-oriented neural networks outperform clustering and autoencoders in solving the uncontrolled vocaburary problem in social tag space. So, due to their repeated success in solving the sparsity, ambiguity, redundancy, and high-dimensionality problems in both the social tag space and the Wikipedia-based feature space, it is believed that the proposed recommendation-oriented neural networks in this thesis can be further used to address similar problems in other topic feature models, e.g., bag-of-words (BoW), latent Dirichlet allocation (LDA), and probabilistic latent semantic analysis (pLSA) topic models.

In addition, we also note that DLSA, ALSA and CLSA generally achieve more significant improvements to ELSA at new locations than those at "old" locations. For example, the MRR of DLSA, ALSA and CLSA are respectively 737.9%, 269.5%, and 197.1% higher than that of ELSA on the New City test subset; while the corresponding improvements are only 464.2%, 128.3%, and 56.6% higher on the Old City test subset. This may be because, due to lack of users' history data on new locations, the inferred localized location profiles on the New City test subset are less accurate than those on the Old City test subset, so ELSA is less likely to recommend users' local target news to the top positions of recommendation lists at new locations. However, by using the clustering or deep learning techniques for topic feature modelling, the proposed geographical topic feature models, CLSA, ALSA and DLSA, map the Wikipedia-based topic space to a more abstract feature topic space, where the correlations among similar topics (e.g., topics within a category) are strengthened. Consequently, CLSA, ALSA and DLSA can uncover users' latent localized news preferences to make the inferred abstract localized location profiles more accurate.

Table 5.4: Online recommendation time-costs of ELSA, CLSA, ALSA, and DLSA (in mins)

	ELSA	CLSA	ALSA	DLSA
The whole test set	341.3	50.56	**12.48**	12.72
Old City test subset	260.8	33.09	**9.437**	9.613
New City test subset	75.51	15.42	**2.293**	2.356

Efficiency and scalability

News recommendation requests real-time responses in practice; so the recommendation efficiency is crucial for online location-aware news recommendation. However, due to the high volume of concepts in Wikipedia, the Wikipedia-based topic space in ELSA is very huge. Consequently, the online news recommendation process that requests to compute similarities between the localized location and news profiles in ELSA is usually computationally expensive. As shown in Table 5.4, even if the Wikipedia topic space has been limited to contain only $8,000$ most frequent topics, the time-costs for the online recommendation processes in ELSA are still up to 341.3, 260.8 and 75.51 minutes on the Old City, the New City and the whole test (sub)sets, respectively, which are usually unacceptable in practice.

Therefore, to ensure scalability in the real-world context, the proposed DLSA (resp., CLSA and ALSA) solve the high-dimensionality problem by mapping the Wikipedia-based topic space to a deep (resp., clusering-based and autoencoder-based) topic feature space with much lower dimensionality. As shown in Table 5.4, although DLSA and ALSA has to pass data through the well-trained neural networks prior to compute similarities between deep or autoencoder-based localized location and news profiles, the online recommendation time-costs of DLSA and ALSA on the whole test set are still only about $1/26$ of that of ELSA. In addition, although the time-cost of CLSA is

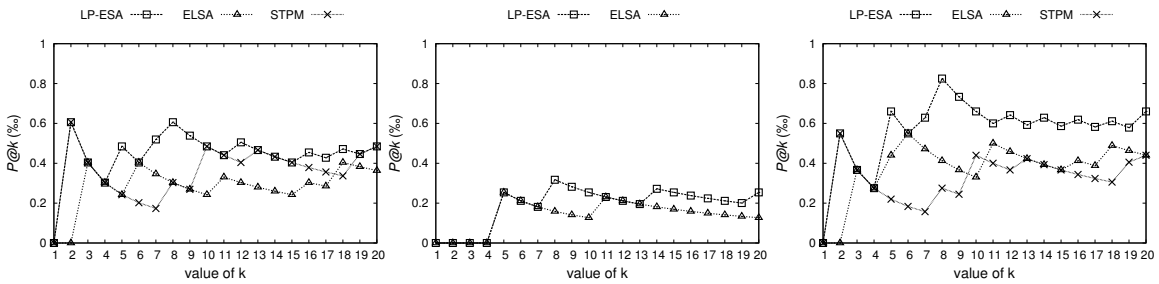
higher than the one of DLSA and ALSA, it is still much lower than that of ELSA. Overall, these findings proves that, with the help of using clustering, autoencoders, and deep neural networks for topic feature modelling, the proposed geographical topic feature models, CLSA, ALSA and DLSA, can greatly reduce the dimensionality of the topic feature spaces; so they can achieve much higher online recommendation efficiency than ELSA and are more scalable in practice.

5.5.3 Effect of Considering Personal Interests

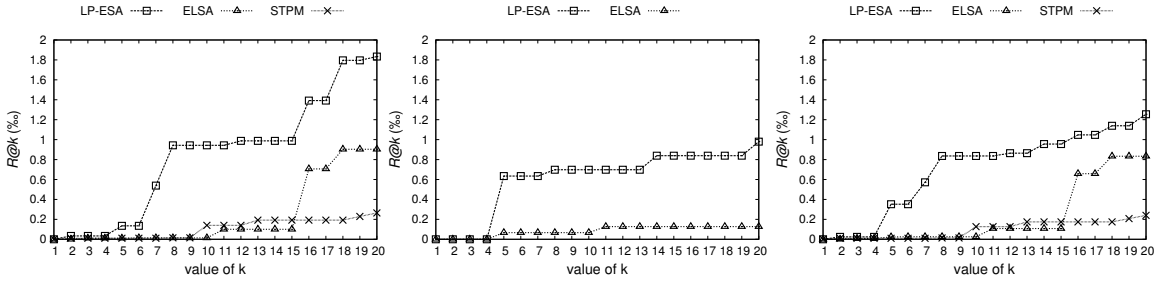
In this subsection, the effect of considering users' personal interests in location-aware news recommendation is evaluated. Therefore, besides ELSA, we also compare LP-ESA with a model called *spatial topical preference model* (STPM) [116], which, to our knowledge, is the only geographical topic-based news recommendation system that considers both the locations and personal interests of users. In STPM, instead of using ESA, the topic model used in STPM is latent Dirichlet allocation (LDA) [24].

Figure 5.10(a)-(i) depicts the news recommendation effectiveness of LP-ESA, ELSA and STPM on three test (sub)sets (i.e., Old City test subset, New City test subset, and the whole test set) in terms of precision at k ($P@k$), recall at k ($R@k$), and F1-score at k ($F@k$), where the value of k varies from 1 to 20. Also, the results of mean reciprocal rank (MRR) of these models on three test (sub)sets are shown in Figure 5.10(j)-(l).

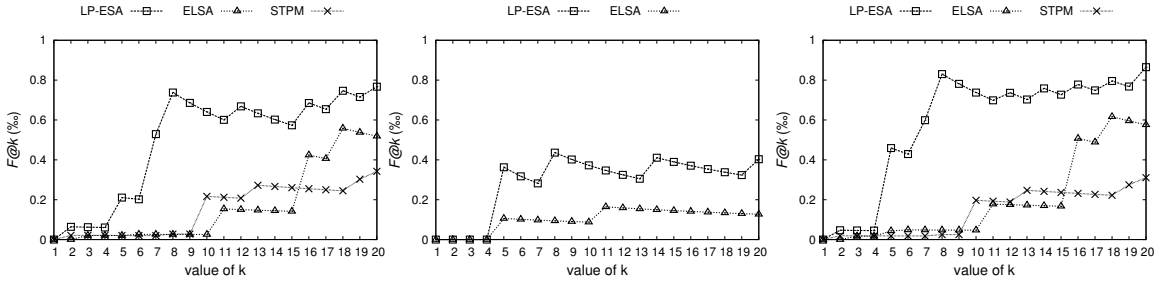
Generally, as shown in Figure 5.10, LP-ESA significantly outperforms the state-of-the-art baselines, ELSA and STPM, in terms of all metrics. Furthermore, we note that there is no results of STPM in Figure Figure 5.10(b), (e), and (h); this is because STPM is inapplicable to the New City test subset: given a user at a location, STPM requests the user's history data on this specific location to model localized news preferences; however, for each sample in the New City test subset, there is no sample in the



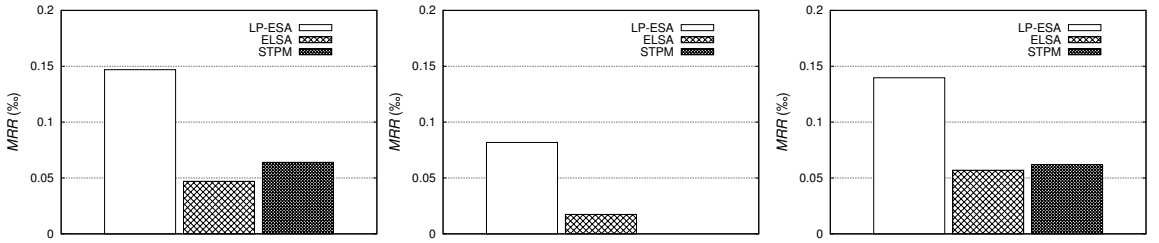
(a) $P@k$ on Old City test subset (b) $P@k$ on New City test subset (c) $P@k$ on the whole test set



(d) $R@k$ on Old City test subset (e) $R@k$ on New City test subset (f) $R@k$ on the whole test subset



(g) $F@k$ on Old City test subset (h) $F@k$ on New City test subset (i) $F@k$ on the whole test set



(j) MRR on Old City test subset (k) MRR on New City test subset (l) MRR on the whole test set

Figure 5.10: The news recommendation effectiveness of LP-ESA, ELSA and STPM on three test (sub)sets in terms of precision at k ($P@k$), recall at k ($R@k$), F1-score at k ($F@k$) and mean reciprocal rank (MRR), with scales on y-axis in ‰

Table 5.5: Average $P@k$, $R@k$, $F@k$ and MRR of LP-ESA, ELSA and STPM on three test (sub)sets when $1 \leqslant k \leqslant 20$ (in ‰)

	Old City test subset			
	P@k	R@k	F@k	MRR
STPM	0.3570	0.1080	0.1504	0.0638
ELSA	0.2834	0.2361	0.1687	0.0470
LP-ESA	**0.4443**	**0.8420**	**0.4919**	**0.1469**
	New City test subset			
	P@k	R@k	F@k	MRR
STPM	\	\	\	\
ELSA	0.1385	0.0836	0.1009	0.0174
LP-ESA	**0.1895**	**0.6053**	**0.2869**	**0.0818**
	The whole test set			
	P@k	R@k	F@k	MRR
STPM	0.3240	0.0980	0.1365	0.0621
ELSA	0.3773	0.2263	0.1981	0.0569
LP-ESA	**0.5675**	**0.6980**	**0.5776**	**0.1397**

training set sharing both the same location and same user with it; so STPM is unable to construct the localized news preferences for users in the New City test subset and, consequently, unable to conduct news recommendation on the New City test subset.

Evaluation and Discussion

In order to evaluate the methods' general performance in news recommendation, according to the values in Figure 5.10, we calculate the average $P@k$, $R@k$, and $F@k$ of these methods on three test (sub)sets when $1 \leqslant k \leqslant 20$, which are then summarized, together with their numerical values of MRR, in Table 5.5.

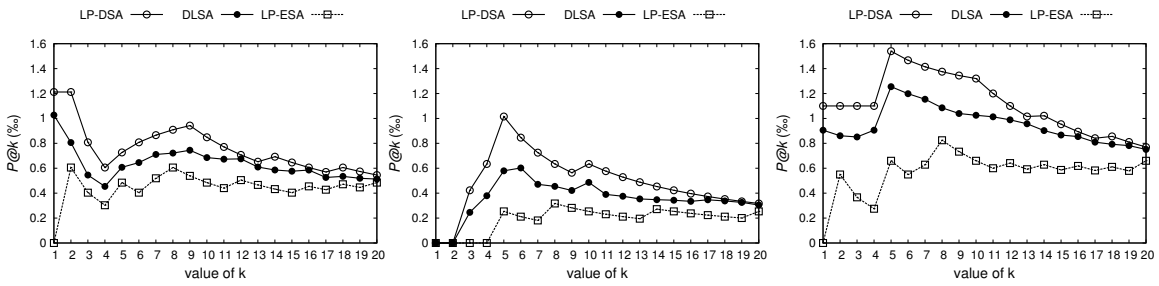
As shown in Figure 5.10 and Table 5.5, LP-ESA outperforms ELSA in all cases, e.g., the MRR of LP-ESA is 2.1, 3.7 and 1.5 times higher than those of ELSA on the Old City test subset, the New City test subset and the whole test set, respectively,

while the comparative results in $P@k$, $R@k$ and $F@k$ are also similar. These results thus show the superiority of LP-ESA, prove the importance of taking into account users' personal interests for location-aware news recommendation, and greatly support our assertion that not only the real-time geographical contexts but also long-term personal interests of users should be considered to achieve a satisfactory real-time news recommendation.

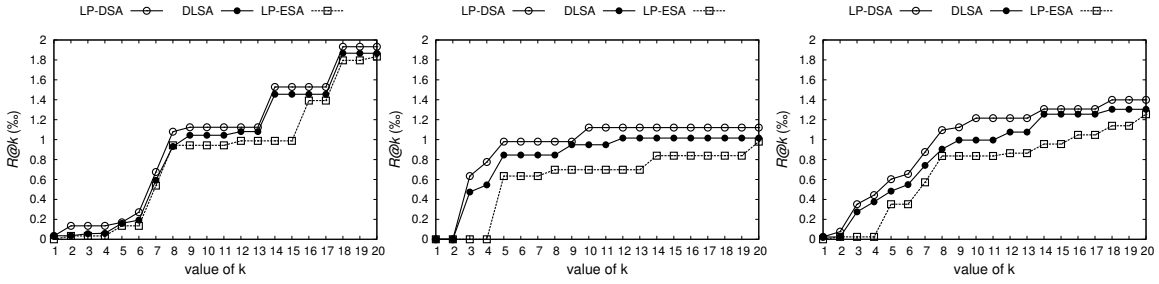
Furthermore, LP-ESA also achieves a much better news recommendation performance than STPM: the average values of $P@k$, $R@k$, $F@k$ and MRR of LP-ESA on Old City test subset are respectively 24.5%, 679.6%, 227.1% and 129.9% higher than those of STPM. The superior performance of LP-ESA may come from the following two aspects: (i) ESA, the topic model used in LP-ESA, which uses Wikipedia for semantic enrichment prior to topic modelling, is more effective in geographical topic-based news recommendation than the LDA topic model used in STMP, especially for the short texts on the Web, e.g., tweets. (ii) Instead of using local history data of users to model localized user preferences directly as in STPM, LP-ESA uses the users' entire history data to model general user profiles and then uses the local topic distribution to localize the general user profiles for location-aware recommendation; consequently, LP-ESA not only overcomes the problem in STPM, i.e., is applicable for conducting news recommendation at new locations, but also achieves a more effective recommendation, as using the entire history data provides a more comprehensive information for modelling users' personal interests.

5.5.4 Performance of LP-DSA

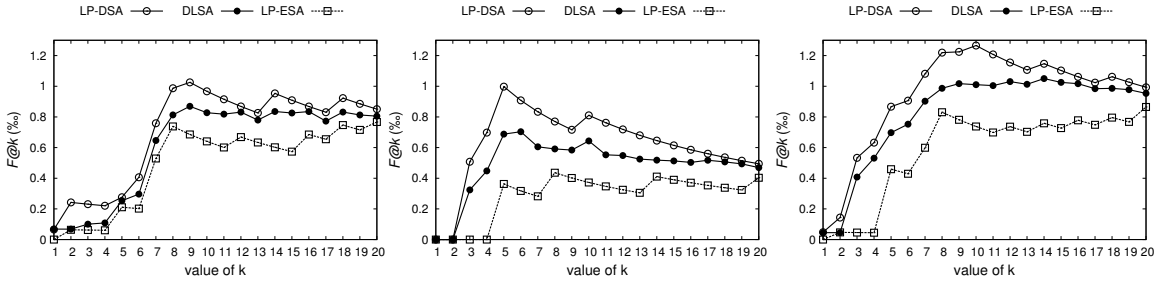
Finally, we evaluate the performance of the proposed hybrid model, LP-DSA, which integrates the ideas of DLSA and LP-ESA and is expected to have the advantages of



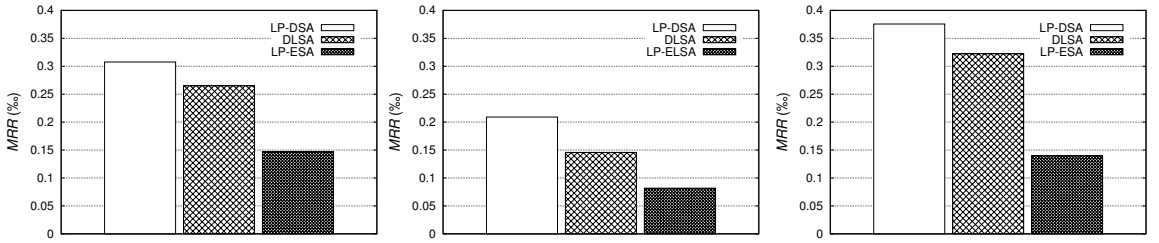
(a) $P@k$ on Old City test subset (b) $P@k$ on New City test subset (c) $P@k$ on the whole test set



(d) $R@k$ on Old City test subset (e) $R@k$ on New City test subset (f) $R@k$ on the whole test subset



(g) $F@k$ on Old City test subset (h) $F@k$ on New City test subset (i) $F@k$ on the whole test set



(j) MRR on Old City test subset (k) MRR on New City test subset (l) MRR on the whole test set

Figure 5.11: The news recommendation effectiveness of LP-DSA, DLSA, and LP-ESA on three test (sub)sets in terms of precision at k ($P@k$), recall at k ($R@k$), F1-score at k ($F@k$) and mean reciprocal rank (MRR), with scales on y-axis in ‰

Table 5.6: Average $P@k$, $R@k$, $F@k$ and MRR of LP-DSA, DLSA, and LP-ESA on three test (sub)sets when $1 \leqslant k \leqslant 20$ (in ‰)

	Old City test subset			
	P@k	R@k	F@k	MRR
LP-ESA	0.4443	0.842	0.4919	0.1469
DLSA	0.6374	0.9389	0.6070	0.2652
LP-DSA	**0.7656**	**1.0083**	**0.7004**	**0.3076**
	New City test subset			
	P@k	R@k	F@k	MRR
LP-ESA	0.1895	0.6053	0.2869	0.0818
DLSA	0.3551	0.8193	0.4867	0.1458
LP-DSA	**0.486**	**0.932**	**0.6176**	**0.2090**
	The whole test set			
	P@k	R@k	F@k	MRR
LP-ESA	0.5675	0.698	0.5776	0.1397
DLSA	0.9500	0.8723	0.8223	0.3224
LP-DSA	**1.1162**	**0.9768**	**0.9403**	**0.3756**

both models. Therefore, in this subsection, we use DLSA and LP-ESA as the baselines and compare the performance of LP-DSA with these two models.

The news recommendation performances of LP-DSA, DLSA, and LP-ESA on three test (sub)sets in terms of precision at k ($P@k$), recall at k ($R@k$), and F1-score at k ($F@k$) with $1 \leqslant k \leqslant 20$ is shown in Figure 5.11(a)-(i). The mean reciprocal rank (MRR) of DLSA, ALSA, CLSA and ELSA on three test (sub)sets are also shown in Figure 5.11(j)-(l). In addition, the average $P@k$, $R@k$, and $F@k$ of LP-DSA, DLSA, and LP-ESA on three test (sub)sets with $1 \leqslant k \leqslant 20$ are computed and summarized, together with their numerical values of MRR, in Table 5.6.

Generally, as shown in Figure 5.11 and Table 5.6, the recommendation performance of LP-DSA greatly outperforms DLSA and LP-ESA in terms of all metrics on all three test (sub)sets. This supports our conclusion that by integrating the ideas of DLSA and

LP-ESA, LP-DSA is capable to take the advantages of both DLSA and LP-ESA and achieves much better performance in location-aware news recommendation.

Specifically, as the deep learning extension of LP-ESA, the experimental results in Figure 5.11 and Table 5.6 show that LP-DSA greatly enhances the performance of LP-ESA on all test (sub)sets and in terms of all metrics. For example, on the whole test set the average $P@k$, $R@k$, $F@k$, and MRR of LP-DSA are respectively 96.7%, 39.9%, 62.8% and 168.9% higher than those of LP-ESA. The improvement is mainly because: (i) By utilizing deep neural networks to extract and compress the incoming Wikipedia-based topic features layer by layer and finally map them to a much denser abstract feature space with much lower dimensionality, LP-DSA can overcome the huge dimensionality, sparsity, and redundancy problems in LP-ESA. (ii) The deep neural networks in LP-DSA are trained with a recommendation-oriented objective to distinguish the user's local target news from the irrelevant ones. Consequently, the resulting abstract features for user, location, and news profiles are very effective representations for location-aware personalized news recommendation.

We also note that the average $P@k$, $R@k$, $F@k$ and MRR of LP-DSA are 156.5%, 54.0%, 115.3% and 155.5% higher than those of LP-ESA on the New City test subset; while the corresponding improvements are only 72.3%, 19.8%, 42.3% and 109.4%, respectively, on the Old City test subset. This observation is consistent with and is another good supporting evidence of our conclusion in Section 5.5.2 that applying deep neural networks for topic feature modelling can uncover users' latent preferences to make the inferred abstract localized user profiles more accurate and remedy the problem of lacking history data at new locations to a great extent.

Furthermore, as the personalized extension of DLSA, the average LP-DSA also outperforms DLSA on all test (sub)sets and in terms of all metrics. For example, the

Table 5.7: Online Recommendation time-costs of ELSA, LP-ESA, DLSA, and LP-DSA (in mins)

	ELSA	LP-ESA	DLSA	LP-DSA
The whole test set	341.3	341.3	**12.72**	12.98
Old City test subset	260.8	260.7	**9.613**	9.781
New City test subset	75.51	75.62	**2.356**	2.379

MRR of LP-DSA is 16.0%, 43.4%, and 16.5% times higher than those of DLSA on the Old City test subset, the New City test subset and the whole test set, respectively, while the comparative results in $P@k$, $R@k$ and $F@k$ are also similar. This again proves our assertion that it is beneficial to take into account users' personal interests for location-aware news recommendation.

Finally, we compare the online recommendation efficiency of ELSA, LP-ESA, DLSA and LP-DSA. As shown in Tabel 5.7, the recommendation time-costs of LP-ESA are very similar to those of ELSA, while the time-costs of LP-DSA are also very close to those of DLSA. This finding indicates that considering users' personal interests does not degrade the recommendation efficiency of location-aware new recommender systems. Also, the online recommendation process of LP-DSA on the whole test set is roughly 25 times faster than that of LP-ESA. This is because, by applying deep neural networks for topic feature modelling, the dimensionality of topic feature space in LP-DSA becomes much lower than LP-ESA, which results in much higher online recommendation efficiency than LP-ESA; this is also consistent with our findings in Section 5.5.2.

CHAPTER 6

Ontology-Based Personalized Recommender Systems

To alleviate the need for powerful computing facilities to train deep recommendation models, in this chapter, we further leverage the ontology techniques [58] to propose ontology-based personalized recommender systems which utilize a novel similarity metric, called *ontological similarity*, to tackle tag ambiguity and redundancy problems without the need of model training. The novelty of the ontological similarity is to use concept similarities in ontologies to estimate the semantic similarities of tags in profiles, which are semantically more accurate than those generated by 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, it is a complement to deep-learning-based recommendation solutions, and will be a good disambiguation choice for light-weight recommender systems.

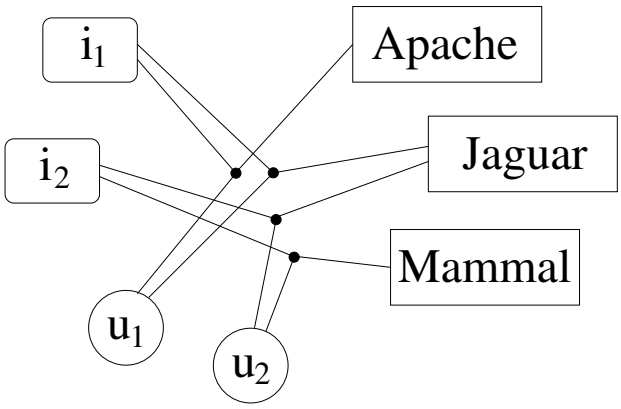


Figure 6.1: Example of a folksonomy

In content-based tag-aware personalized recommender systems, 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 items (described by a item profile), where the vector space model (VSM) [138] is used to represent both profiles as weighted vectors of tags. To calculate the similarity between the weighted vectors of these two profiles, the most widely adopted measure is the *cosine similarity* [32, 178]. In addition, Vallet et al. [164] propose the *scalar similarity* to try to improve the metric by eliminating the vector length normalization factors in the 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 items among users, and normalization thus penalizes popular items.

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 items usually contain much ambiguous or redundant information, e.g., homonyms (tags with the same spelling but of different meaning) or synonyms (different tags of the same meaning). Thus, performing a literal matching

over such ambiguous or redundant tags may lead to inaccurate similarity values between corresponding profiles, which greatly degrade the performance of the content-based recommendation system. For example, we recall the folksonomy first given in Chapter 3 and re-depict it as Figure 6.1; in this folksonomy, 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" (if used as tags) are similar in semantics, but their cosine and scalar similarity are 0 (irrelevant).

Consequently, to tackle the tag ambiguity and redundancy problems and achieve better performance in content-based tag-aware personalized recommendation, we thus propose a novel similarity measure, called *ontological similarity*, which 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.

Specifically, in this work, we propose a two-step top-down disambiguation algorithm to address the tag ambiguity and redundancy problems by mapping tags to unique matching concepts in the ontology: Given either a user or an item profile, we first map tags in this profile to all the possible concepts in the ontology (called *mapping step*); however, due to homonyms, it may be possible to map a certain tag to multiple concepts (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 statistic information of matching concepts of other tags in the same profile as context to disambiguate tags with more than one matching concept (called

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 (resp., redundancy) 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).

The rest of this chapter is organized as follows. I first describe the new ontological similarity measure in Section 6.1. Then, Section 6.2 presents the personalized recommendation based on this ontological similarity. Finally, Sections 6.3 provides the experimental results.

6.1 Ontological Similarity

However, the two state-of-the-art similarity measures apply literal matching of tags; due to ambiguous and redundant 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, which uses ontologies to disambiguate tags and leverages concept similarities 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 similarities of profiles are 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[1] as an example of the 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 [61]. However, the methods proposed here are not restricted to ODP and can be used with other ontologies, e.g., WordNet.

6.1.1 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 a most relevant concept as the matching concept of this tag. One existing method [9] is to first map the given tag to all possible candidate concepts and then use the contextual information of the profile to identify a most relevant concept: it estimates the similarities of all combinations of this tag and one another tag in this profile by using the Wu and Palmer concept similarity [175] in the ontology; then the candidate concept that re-

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

turns the highest Wu and Palmer similarity is selected as the matching concept. 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.

However, as the first step for computing the ontological similarity, tag allocation is invoked very frequently in ontology-based recommendation; therefore, the high computational complexity of the existing tag allocation solution [9] will make the recommendation process inefficient and time-consuming. Consequently, in this work, we propose a more efficient two-step top-down traversal disambiguation method, whose complexity is $O(m \cdot r^2)$ (or $O(m \cdot r \cdot \log(r))$, if the ontology is a balanced tree) in the worst case. Similar to [9], 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 and incrementally narrow down the number of candidate concepts by using the statistic information of the matching concepts of other tags in the same profile as context.

The algorithm of the top-down traversal tag allocation is provided in Algorithm 2. Intuitively, the high efficiency of the proposed top-down traversal solution comes from its capability to utilize the tree structure of the taxonomy to ensure the matching concept can be found by following only the highest weighted branch of the tree and incrementally removing the candidate concepts belonging to other branches in the disambiguation step. This solution can be seen as a pruning algorithm, and its detailed process is described as follows.

Mapping step (Lines 4–11): given a profile, for each tag, we traverse the whole

ALGORITHM 2: TagAllocation(x, *Ont*)

Input: profile (either user or item profile): x, domain ontology: *Ont*
Output: $hashmap_{match}$ — a hashmap saving the matching concepts of tags in x

```
1  foreach concept c ∈ Ont do
2      cw_c = 0;                                   // Initialize the concept weight of c, denoted cw_c
3  hashmap_match = ∅;
4  foreach (tag t, weight w_t) ∈ x do
5                                                  // Mapping step
6      list_t = ∅;
7      foreach concept c ∈ Ont that matches t do
8          list_t.add(c);
9          UpdateConceptWeights(c, w_t, Ont);      // Algorithm 3
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) ∈ x 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' ∈ ChildrenOf(c_s) do
19             if DescendantsOf(c') ⋂ list_t ≠ ∅ then
20                 if cw_c' < MaxWeight then
21                     list_t.remove({DescendantsOf(c') ⋂ list_t)
22                 else
23                     if MaxConcept ≠ null then
24                         list_t.remove({DescendantsOf(MaxConcept) ⋂ list_t);
25                     MaxWeight = cw_c';
26                     MaxConcept = c';
27         c_s = MaxConcept                        // update c_s to the max child concept
28     while SizeOf(list_t) > 1;
29     hashmap_match.add(t, list_t[0]);  // set the only concept left in list_t as matching concept of t
30 return hashmap_match.
```

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

ALGORITHM 3: UpdateConceptWeights(c, w, *Ont*)

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

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), these concepts are marked as *candidate matching concepts* and eliminate its ambiguity later. To help disambiguation, we increase the concept weights (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 traversal 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 the child concept with the highest concept weight as the new c_s (Line 27) and unmark the candidate concepts that are not the descendants of the new c_s (Lines 20–21 and 23–24); 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).

The following example illustrates the top-down traversal 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 6.1, and let u be a user who is a military enthusiast and has a user profile x_u containing two tags, *Apache* and *Jaguar*, with weights 5 and 3, respectively—i.e., $x_u = \{(Apache, 3), (Jaguar, 5)\}$.

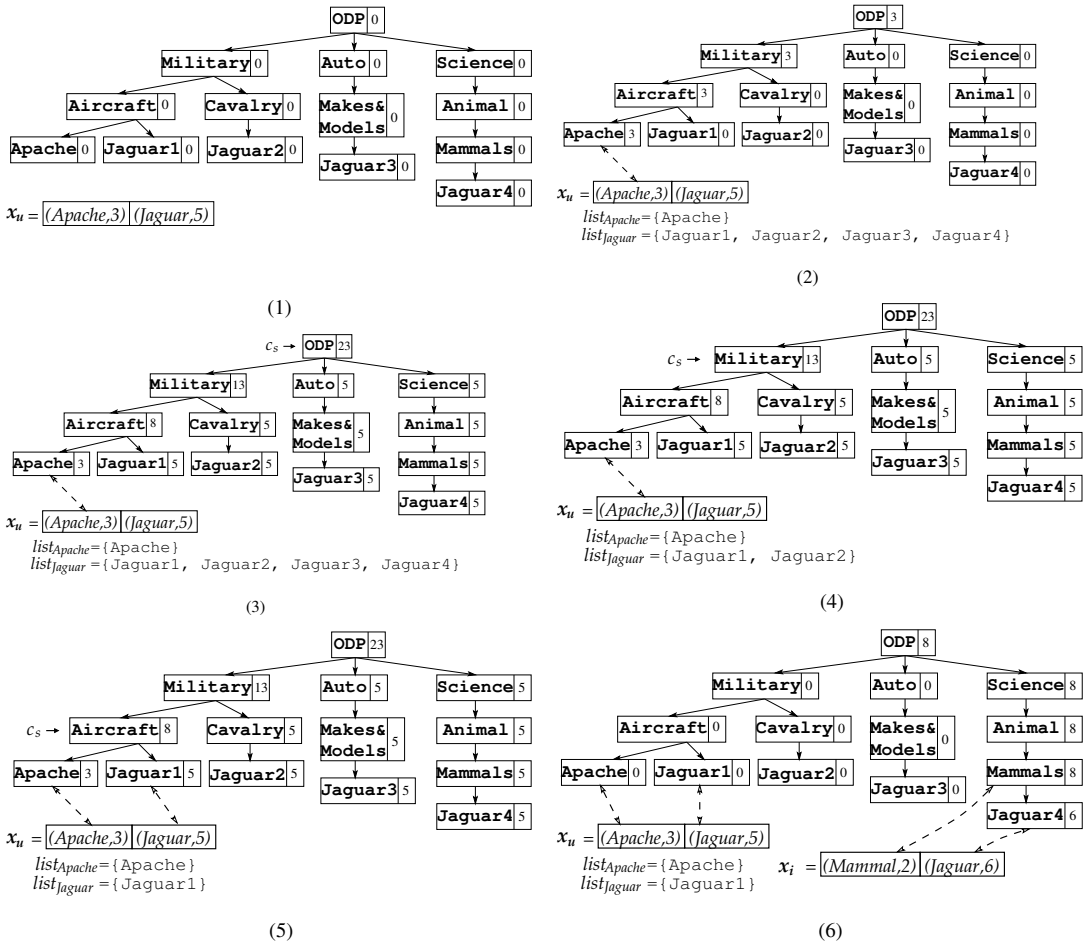


Figure 6.2: Allocation of tags in the ODP taxonomy

We would like to allocate these two tags to concepts in the ODP ontology as shown in Figure 6.2. According to Algorithm 2, we first initialize all concept weights in the ontology to 0 (as shown in Figure 6.2 (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 6.2 (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 the list $list_{Jaguar}$ is larger than 1. Consequently, we increase the concept weights of all *Jaguar* concepts and their ancestors by 5 and invoke the disambiguation procedure to solve the multiple occurrence problem as 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 6.2 (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 6.2 (4)). As there are still two distinct candidate concepts (*Jaguar1* and *Jaguar2*) in $list_{Jaguar}$, the do-while-loop in Algorithm 2 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 6.2 (5)).

Similarly, given an online item i about animals, with profile $x_i = \{(Mammal, 2), (Jaguar, 6)\}$, 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 x_i (as shown in Figure 6.2 (6)). This example shows the advantage of the proposed tag allocation methodology in dynamically allocating tags to the semantically most suitable concepts. ∎

The following shows the worst case complexity of top-down traversal tag allocation in Algorithm 2 in general cases and when the underlying taxonomy is a balanced tree. Note that it is rare to have very unbalanced tree structure in the taxonomy in the real-world context so the practical computational cost is usually closer to the balanced tree case.

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

They are, however, 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 2 is thus $O(m \cdot r^2)$ (resp., $O(m \cdot r \cdot \log r)$). □

6.1.2 Computation of Ontological Similarity

After the tag allocation, it is possible to compute the ontological similarity between a given user profile and a given item profile. For a matching concept c_u of a user profile tag t_u, we first define its *nearest concept* c_i as the matching concept of a tag t_i in x_i that satisfies the following two conditions: (i) the least common ancestor (*lca*) of c_u and its nearest concept c_i must be a descendant of (or the same as) the *lca* of c_u and any other matching concept of a tag in x_i; and (ii) if there exist other matching concepts of a tag in x_i whose *lca* with c_u is the same as that of c_i, these concepts must be in a lower level of the hierarchy than (or the same level as) c_i. The intuition behind this

definition is that concept specifications in a taxonomy are recursively refined, so the defined nearest concept c_i is the semantically closest one to c_u compared to any other matching concept of a tag in x_i.

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_i of a tag t_i to estimate the semantic similarity of t_u and t_i. This similarity measure is called *concept similarity* and we 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.

The process of computing the cosine-based ontological similarity is summarized in Algorithm 4. Generally, given a user profile x_u and a item profile x_i, this algorithm estimates the semantic similarity between the two given profiles by using the concept similarity between the matching concepts of the tags in x_u and their nearest concepts in x_i. 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 (resp., high) ontological similarity between homonyms (resp., synonyms) in different profiles.

Each step of Algorithm 4 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) *Computing Concept Similarity (Lines* 4–26*):* For each tag t_u with a weight w_u in x_u, we check whether it has a matching concept c_u in *Ont*, and whether at least

ALGORITHM 4: OntologicalSimilarity$\big(x_u, x_i, \mathit{hashmap}_{x_u}, \mathit{hashmap}_{x_i}, \mathit{Ont}\big)$

foreach *concept* $c \in \mathit{Ont}$ **do**
  $\mathit{counter}_c = 0$; // counts the number of times c is selected as the nearest concept
$\mathit{sum}_u = 0$; $\mathit{sum}_i = 0$; $\mathit{Similarity} = 0$;
foreach $(t_u, w_u) \in x_u$ **do**
  $\mathit{sum}_u \mathrel{+}= (w_u)^2$; $c_u = \mathit{hashmap}_{x_u}.get(t_u)$; // w_u is the weight of tag t_u in x_u, and $c_u \in \mathit{Ont}$ is the matching concept of t_u
  if $c_u \neq null$ *and* $SizeOf(\mathit{hashmap}_{x_i}) > 0$ **then**
    $\mathit{lca} = c_u$; *lca'* = `null`; $c_i =$ `null`; // *lca* is found bottom-up from c_u to root
    do
      foreach *concept* $c' \in \mathit{Subtree}(\mathit{lca})$ *with* $c' \notin \mathit{Subtree}(\mathit{lca'})$, *in breadth-first exploration* **do**
        if c' *is the matching concept of a tag* t_i *with* $(t_i, w_i) \in x_i$ **then**
          // w_i is the weight of tag t_i in x_i
          $c_i = c'$;
          $\mathit{counter}_{c_i}{+}{+}$;
          break;
      if $c_i =$ `null` **then**
        lca' = *lca*;
        $\mathit{lca} = \mathit{Parent}(\mathit{lca})$;
    while $c_i =$ `null`;
    conSim = $\mathit{ConSim}(c_u, c_i)$ according to Equations 6.1 or 6.2
  else
    // when t_u is unmatched tag, or no tag in x_i has matching concept
    if $t_u = t_i$ *for a* $(t_i, w_i) \in x_i$ **then**
      conSim $= 1$
    else
      conSim $= 0$;
  $\mathit{Similarity} \mathrel{+}= w_u \cdot w_i \cdot \mathit{conSim}$;
foreach $(t_i, w_i) \in x_i$ **do**
  $k = 1$;
  if $\mathit{counter}_{c_i} > 1$ **then**
    $k = \mathit{counter}_{c_i}$;
  $\mathit{sum}_i \mathrel{+}= k \cdot (w_i)^2$;
$\mathit{ontSim}_{Cosine} = \frac{\mathit{Similarity}}{\sqrt{\mathit{sum}_u} \cdot \sqrt{\mathit{sum}_i}}$; // according to Equation 6.3
return ontSim_{Cosine}.

one tag in x_i has a matching concept (Line 6). If not, we set the concept similarity to 1, if t_u is also in x_i, 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_i and lowest common ancestor concept (*lca*), and then compute the concept similarity of c_u and c_i, using the measures defined in Equation 6.1 (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 x_i. If so, we select c' as the nearest concept c_i of c_u, increase $counter_{c_i}$ by one, and stop the search (Lines 9–14). To make sure that the first found matching concept of x_i 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_i 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_i is found (Line 18).

(3) *Computing Ontological Similarity (Lines 27–32):* Finally, the ontological similarity is computed according to the proposed metric in Equation 6.3, 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 4, 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 (see Equation 6.4).

The following result states the cost of computing ontological similarities with Algorithm 4 (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 item profile of size m_i, Algorithm 4 runs in time $O(m_u \cdot (r + m_i))$ 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 x_i in Line 22 needs $O(m_i)$. Finally, the loop in Line 27 takes time $O(m_i)$. The total worst-case running time is therefore $O(m_u \cdot (r + m_i))$. □

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 [52, 102, 124]. However, the use of the proposed technique is not restricted to these two path-based metrics; other metrics, such as the information-content-based Jiang-Conrath similarity [76] and the Lin similarity [104] 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 [52, 175] can be used to define the similarity of two concepts c_u and c_i, namely:

$$\textit{ConSim}_1(c_u, c_i) = \frac{2 \cdot l(\textit{lca}(c_u, c_i))}{l(c_u) + l(c_i)}, \tag{6.1}$$

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

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 [132]. Formally, for two concepts c_u and c_i:

$$\textit{ConSim}_2(c_u, c_i) = \frac{1}{\textit{SP}(c_u, c_i) + 1}, \tag{6.2}$$

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

Example 2 Continuing Example 1, the concept similarities for the tags in x_u and x_i 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 $l(\textit{root}) = 0$ and $l(\textit{Jaguar1}) = l(\textit{Mammals}) = 3$, the concept similarity in Equation 6.1 is $\textit{ConSim}_1(\textit{Jaguar1}, \textit{Mammals}) = \frac{2 \cdot 0}{3+3} = 0$. Similarly, $\textit{ConSim}_1(\textit{Apache}, \textit{Mammals}) = 0$. The concept similarity in Equation 6.2 for these two cases can be computed analogously. ∎

Ontological Similarity

Finally, the ontological similarity between two profiles x_u and x_i 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. Formally, the cosine-based ontological similarity is defined as follows:

$$\mathit{ontSim}_{\mathit{Cosine}}(x_u, x_i) = \frac{\sum_{j=1}^{|T_u|}(w_{u_j} \cdot w_{i_j} \cdot \mathit{ConSim}(c_{u_j}, c_{i_j}))}{\sqrt{\sum_{j=1}^{|T_u|}(w_{u_j})^2} \cdot \sqrt{\sum_{nc=1}^{|T_i|} k_{nc} \cdot (w_{i_{nc}})^2}}, \tag{6.3}$$

where $|T_u|$ and $|T_i|$ are the numbers of different tags in x_u and x_i, respectively; c_{u_j} is the matching concept of a tag t_{u_j} in x_u; c_{i_j} is c_{u_j}'s nearest concept mapped by t_{i_j} in x_i; and w_{u_j}, w_{i_j}, and $w_{i_{nc}}$ are the weights of tags t_{u_j}, t_{i_j}, and $t_{i_{nc}}$, respectively. As some concepts in x_i may be used multiple times as the nearest concept of different concepts in x_u (e.g., *Mammals* is used for both *Apache* and *Jaguar1* in Example 2), k_{nc} is the number of times that $t_{i_{nc}}$'s matching concept is used as the nearest concept, if $t_{i_{nc}}$ is a multi-selected tag; otherwise, $k_{nc} = 1$ (see Lines 27–29 in Algorithm 4). $\mathit{ConSim}(c_{u_j}, c_{i_j})$ is the concept similarity of c_{u_j} and c_{i_j} as predefined, if t_{u_j} has a matching concept; if t_{u_j} is unmatched, then $\mathit{ConSim}(c_{u_j}, c_{i_j})$ is 1, if the same tag is in x_i, and 0, otherwise.

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

$$\mathit{ontSim}_{\mathit{Scalar}}(x_u, x_i) = \sum_{j=1}^{|T_u|}(w_{u_j} \cdot w_{i_j} \cdot \mathit{ConSim}(c_{u_j}, c_{i_j})), \tag{6.4}$$

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 6.1 for concept similarity. Then, it is straightforward to compute the cosine-based ontological similarity between x_u and x_i as $\mathit{ontSim}_{\mathit{Cosine}}(x_u, x_i) = 0$, because $\mathit{ConSim}_1(\mathit{Jaguar1},$

Mammals) = *ConSim*$_1$(*Apache*, *Mammals*) = 0.

However, given a new online item i' regarding a military organization with $x_{i'} = \{(Military, 10)\}$, as in Examples 1 and 2, we obtain *ConSim*$_1$(*Jaguar1*, *Military*) = *ConSim*$_1$(*Apache*, *Military*) $= \frac{2 \cdot 1}{1+3} = 0.5$. Subsequently, we compute the ontological similarity value between x_u and $x_{i'}$ as follows:

$$\textit{ontSim}_{\textit{Cosine}}(x_u, x_{i'}) = \frac{5 \cdot 10 \cdot 0.5 + 3 \cdot 10 \cdot 0.5}{\sqrt{(5)^2 + (3)^2} \cdot \sqrt{2 \cdot (10)^2}} = 0.485,$$

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, i' will be re-ranked much higher than i, 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: i is ranked higher than i', as x_i shares the tag *"Jaguar"* with x_u. ∎

6.2 Personalized Recommendation Based on Ontological Similarity

Finally, an algorithm that uses the ontological similarity between the user profile and item profiles to achieve personalized recommendation on the Social Web is presented in this section.

In this work, we reasonably assume that the user's preferences (described by the user profile) and the social summaries of items (described by the item 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 item profiles, and then store the resulting similarity values for the eval-

ALGORITHM 5: PrecomputeOntologicalSimilarity($x_u, list_{x_i}, Ont$)

Input: User profile x_u, a list of item profiles $list_{x_i}$, and domain ontology Ont
Output: $list_{ontSim}$ — a list stores the resulting ontological similarity values

1 $hashmap_{x_u} = TagAllocation(x_u, Ont)$; // Algorithm 2
2 **foreach** $x_{i_j} \in list_{x_i}$ **do**
3 $\quad hashmap_{x_i} = TagAllocation(x_{i_j}, Ont)$; // Algorithm 2
4 $\quad Sim_{x_{i_j}} = OntologicalSimilarity(x_u, x_{i_j}, hashmap_{x_u}, hashmap_{x_i} Ont)$; // Algorithm 4
5 $\quad list_{ontSim}.add(Sim_{x_{i_j}})$;
6 return $list_{ontSim}$.

uation of online personalized recommendations. Note that, to adapt to changes of user and item profiles, periodic profile updates can be conducted, and the similarity computation process can be done again.

The pre-computation of ontological similarities is described in Algorithm 5. It takes as *input* a weighted vector denoting a user u's profile (x_u), a list of weighted vectors denoting item profiles ($list_{x_i}$), and an external domain ontology (Ont). This algorithm consists of three steps: first, it allocates (maps) tags in x_u to concepts in Ont (Line 1); then, for each item i_j in the list $list_{x_i}$, it allocates tags in x_{i_j} to concepts in Ont, computes the ontological similarity of x_u and x_{i_j}, 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 5.

Proposition 3 *Given a taxonomy Ont with r concepts, a user profile x_u of size m_u, and a item profile list $list_{x_i}$ of length n, with item profiles x_{i_j} of size m_i, Algorithm 5 runs in time $O(n \cdot m_i \cdot r^2 + n \cdot m_u \cdot (r + m_i) + m_u \cdot r^2)$ in the worst case. If the underlying taxonomy is a balanced tree, then the worst-case complexity of Algorithm 5 is $O(n \cdot m_i \cdot r \cdot \log r + n \cdot m_u \cdot (r + m_i) + m_u \cdot r \cdot \log r)$.*

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 $O(m_i \cdot r^2 + m_u \cdot (r + m_i))$ (resp., $O(m_i \cdot r \cdot \log r + m_u \cdot (r + m_i))$ for the balanced-tree case): specifically, allocating item profile tags in Line 3 takes $O(m_i \cdot r^2)$ (resp., $O(m_i \cdot r \cdot \log r)$), and the computation of the ontological similarity in Line 4 takes $O(m_u \cdot (r + m_i))$ based on Proposition 2. So, the total worst-case running time of Algorithm 5 is $O(n \cdot m_i \cdot r^2 + n \cdot m_u \cdot (r + m_i) + m_u \cdot r^2)$ (resp., $O(n \cdot m_i \cdot r \cdot \log r + n \cdot m_u \cdot (r + m_i) + m_u \cdot r \cdot \log r)$ for the balanced-tree case). □

So, based on Equation 3.1, personalized recommendation can be done by sorting the items 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 the 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 items, whose complexity is $O(n \cdot \log n)$ in the worst case. □

6.3 Experiments

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

Experiments are conducted over a public real-world *Delicious* dataset, which is gathered from the Delicious bookmarking system and is released in HetRec 2011 [34].

Table 6.1: Dataset information

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

To cut down the time of calculation, we conduct a preprocessing to remove infrequent tags, i.e., tags that are used less than 15 times. The statistic information about the resulting dataset is as shown in Table 6.1, where 41.9% of users have less than 50 tags; 31.3% and 22.8% of users have 50-100 and 100-200 tags, respectively; and only 4% of users have more than 200 tags.

Furthermore, as stated previously, we use the ODP taxonomy as the underlying ontology, which contains $802\,456$ categories (i.e., concepts) in total; while $1\,637$ 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.

6.3.1 Performance in Tag Disambiguation

As discussed in Chapter 2, 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 [38] and [110], 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 and redundancy problems, tag-context-based, item-context-based, and user-context-based distributional measures (denoted *tagCont*, *docCont*, and *userCont*, respectively) as defined in [38] are selected as the baselines of the tag disambiguation performance evaluation.

Evaluation Methodology

As distributional measures rely on the most related tags for tag disambiguation, their tag disambiguation performances depend on the semantic distance in 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 [38] and [110]; here, ODP is used as the underlying ontology for semantic grounding, and the 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 [38]: 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 relative depth of concepts and the shortest path (as defined in Equation (6.1) and (6.2)) are used and denoted as $Onto\text{-}based_1$ and $Onto\text{-}based_2$, respectively.

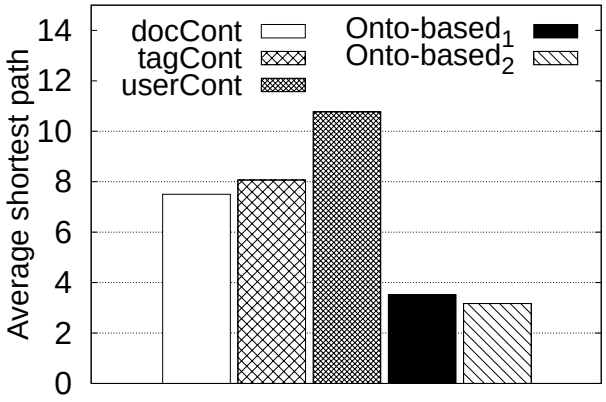


Figure 6.3: Average semantic distance of $docCont$, $tagCont$, $userCont$, $Onto\text{-}based_1$, and $Onto\text{-}based_2$, estimated by the average length in the ODP taxonomy of the shortest paths between the matching concepts of original tags and those of their most related tags

Experimental Results

Figure 6.3 depicts the average semantic distances of $docCont$, $tagCont$, $userCont$, $Onto\text{-}based_1$, and $Onto\text{-}based_2$, estimated by the average lengths of the shortest paths in the ODP taxonomy from the matching concepts of original tags to those of the most related tags. Generally, the shorter the average semantic distance, the better the disambiguation performance.

As shown in Figure 6.3, $Onto\text{-}based_1$ and $Onto\text{-}based_2$ both greatly outperform all the baselines: their average semantic distances are less than half of that of the best baseline, docCont, while the performance of $Onto\text{-}based_2$ is slightly better than $Onto\text{-}based_1$. The superior performance of ontology-based methods is mainly because: for a given tag, the ontology-based methods find its most related tag as the tag mapping to a concept that has the highest concept similarity to the given tag's matching concept in ODP; especially, for $Onto\text{-}based_2$, the selected matching concept of the most related tag is guaranteed to be the one with the shortest path to the given tag's matching con-

cept among all matching concepts in ODP; so, its average shortest path (and semantic distance) is guaranteed to be optimal.

Despite achieving superior performance in tag disambiguation, the ontology-based solution has still a limitation: due to the need of mapping tags to ontologies, it can only disambiguate tags that can be mapped to ontologies. Therefore, although we have considered the similarity between unmatched tags using the cosine similarity (Lines 20-25 in Algorithm 2), it will still be beneficial to integrate the non-ontology-based disambiguation solutions, such as co-occurrence tags and distributional measures [110], 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 the future work.

6.3.2 Performance in Tag-to-Ontology Allocation

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 [9] 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 another tags in this profile using the Wu and Palmer similarity [175] 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.

Evaluation Methodology

To evaluate the tag-to-ontology allocation accuracy, we aim to 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 item 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 as *MASD*), formally defined as follows:

$$\textit{MASD} = \frac{1}{N_p} \sum_{a=1}^{N_p} \frac{1}{N_a^t} \sum_{j=1}^{N_a^t} aveSemDis(c_{a,j}), \tag{6.5}$$

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, same as Section 6.3.1, 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 6.1.1, the

Table 6.2: Accuracy and efficiency of tag-to-ontology allocation

	MASD_i	time_i	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

propose of proposing the top-down traversal tag allocation strategy in this work is to overcome the high computational complexity problem in the existing strategy [9]. 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.

Experimental Results

Table 6.2 shows the mean average semantic distance (MASD) of allocating tags in all item and user profiles (denoted $MASD_i$ 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 $Top\text{-}Down$), as well as their average time-costs for allocating tags for each user and item profile (denoted $time_i$ and $time_u$, respectively).

As shown in Table 6.2, as for tag allocation in item profiles, the $MASD_i$'s of $Wu\,\&\,Palmer$ and $Top\text{-}Down$ are very close (3.708 vs. 4.069), while the time needed for $Wu\,\&\,Palmer$ is 4.9 times as long as that of $Top\text{-}Down$ (0.280 secs vs. 0.057 secs). Similarly, for the user profiles, the $MASD_u$ of $Top\text{-}Down$ is only 7.1% higher than that of $Wu\,\&\,Palmer$ (8.785 vs. 9.415) but its mapping efficiency is about 5.1 times quicker than that of $Wu\,\&\,Palmer$ (2.69 secs vs. 0.44 secs). 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 [9], while main-

taining a very close allocation accuracy. This is also consistent to our computational complexity analysis in Section 6.1.1.

6.3.3 Performance in Personalized Recommendation

Finally but most importantly, we evaluate the performance of the proposed ontological-similarity-based recommendation model in content-based tag-aware personalized recommendation. Therefore, two content-based tag-aware personalized recommender systems based on the widely used state-of-the-art similarity metrics, cosine similarity [178] and scalar similarity [164], are selected as baselines and denoted $Cosine$ and $Scalar$, respectively.

In addition, 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).

Evaluation Methodology

Although the relevance and/or value of personalization depends on users' subjective views, several studies [19, 22, 83] have shown that users' tagging behaviour is closely correlated with their personal relevance judgement; i.e., if a item is annotated by a user with certain tags, this item is very likely to be visited by the same user, if it appears as a recommendation result. This is the basis of the automatic evaluation: the plausible items in personalized recommendation are the ones annotated by a user.

Therefore, we assume that the target items of a given user are those annotated by

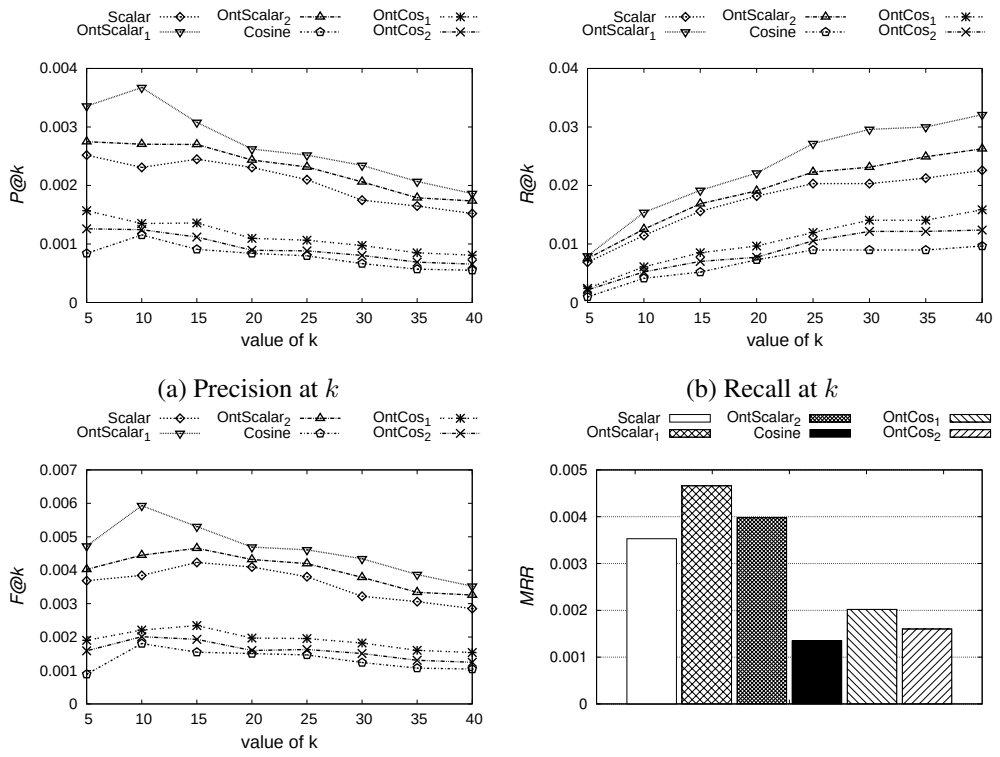
this user. Thus, for each user u annotating more than one item in the dataset, we randomly select at least one item i annotated by this user as his/her target item and put the selected (u, i) pairs in a testing dataset, where we aim at recommending i given u in the testing. Then, for each (u, i) pair in the testing set, we remove from the dataset all the assignments given by u to i; the remaining assignments in the dataset are used as historical dataset which is used to construct user and item profiles for content-based tag-aware personalized recommendations. Finally, we get $3\,529$ user-item pairs in the testing dataset and $274\,362$ assignments in the history dataset. We name the remaining dataset as historical dataset, instead of training dataset, to emphasize that, differently from machine-learning-based solutions, the recommendation solutions presented in this work do not need model training; so the historical dataset is not used for training but only used to construct user and item profiles.

As for the evaluation of the personalized recommendation, we continue to use the *precision at k ($P@k$)*, *recall at k ($R@k$)*, *F*1*-score at k ($F@k$)*, and *mean reciprocal rank (MRR)* as defined in Section 4.3 of Chapter 4.

Results and Evaluations

In this section, we compare the personalized recommendation performances of tag-aware content-based recommender systems using ontological similarities to those using the conventional cosine and scalar similarities.

Figure 6.4 depicts in detail the personalized recommendation performances 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 $k = 5$,



(a) Precision at k

(b) Recall at k

(c) F1-score at k

(d) MRR

Figure 6.4: The recommendation performance of $Scalar$, $OntScalar_1$, $OntScalar_2$, $Cosine$, $OntCos_1$, and $OntCos_2$ on in terms of precision at k ($P@k$), recall at k ($R@k$), F1-score at k ($F@k$), and mean reciprocal rank (MRR)

$10, \ldots, 40$. In addition, to evaluate their general performances, 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 6.3.

Generally, as shown in Figure 6.4 and Table 6.3, 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 [164] that scalar similarity can outperform cosine similarity in content-based tag-aware personalized recommendation by eliminating the profile length normalization factors.

Table 6.3: 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

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 performances 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., $OntScalar_1$ 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 6.1), may be more effective than the shortest-path-based concept similarity $Consim_2$ (as defined in Equation 6.2) in measuring the semantic relevance between matching concepts of tags in profiles, and thus achieves better performance in ontology-based personalized recommendation.

Evaluation vs. HDLPR-NS

The deep-learning-based HDLPR-NS model (as presented in Chapter 4) is also proposed in the context of tag-aware personalized recommendation for disambiguation. However, it is not particularly designed for light-weight recommender systems. To evaluate the pros and cons of the proposed ontological similarity, we further compare it with HDLPR-NS.

We note that the design of experimental studies for HDLPR-NS in Chapter 4 is slightly different to the experimental settings here. In this section, we select only a few user-item testing samples for each user (resulting only 3529 samples in the testing dataset) to simulate the practical usage of light-weight recommender systems. However, in HDLPR-NS, we randomly select 15% of assignments as testing samples to simulate the various usage patterns of different users, resulting various numbers of testing cases for different users (ranging from one or two to tens and even hundreds).

Therefore, we re-run HDLPR-NS using the data partition as shown in section 6.3.3. Too keep fair comparison, we do not construct a validate dataset (otherwise, it will change the data partition) but train the model for 100 runs, which are generally sufficient to reach convergence according to the previous finding in section 4.3.2 (i.e.,

Table 6.4: HDLPR-NS vs. OntScalar_1

	avg. time	$P@k$	$R@k$	$F@k$	MRR
HDLPR-NS	1.83 secs	**0.003315**	**0.03147**	**0.00625**	**0.00645**
OntScalar_1	**0.17 secs**	0.00268	0.02293	0.00462	0.00466

training of HDLPR-NS converge within tens of runs). The performance of HDLPR-NS in terms of average time-cost of a testing case, $P@k$, $R@k$, $F@k$, and MRR, are exhibited in Table 6.4.

As The results in Table 6.4 mainly indicate the following observations: (i) The scalar-based ontological similarity has much lower time-cost for recommendation than HDLPR-NS (0.17 secs vs. 1.78 secs). (ii) With no surprise, the recommendation accuracy of HDLPR-NS is better than OntScalar_1, in terms of $P@k$, $R@k$, $F@k$, and MRR. Therefore, although HDLPR-NS can achieve better recommendation accuracy than OntScalar_1, its computational cost is significantly higher than OntScalar_1. This proves that HDLPR-NS is generally a better choice for tag-aware recommendation, and the proposed ontological similarity is a reasonable disambiguation alternative for light-weight recommender systems without powerful computing capability.

CHAPTER 7

Comparisons and Suggestions

In this chapter, we will compare the proposed recommendation models, HDLPR-NS, LP-DSA, and OntSim, to advise their relative merits and potential real-world applications. The comparisons here are based on four perspectives, application area, personalization, main challenge, and technique used, as listed in Table 7.1

We first compare HDLPR-NS and LP-DSA. Overall, HDLPR-NS is proposed in the context of tag-aware personalized recommendation, while LP-DSA is proposed for geographical topic-based location-aware news recommendation. Therefore, HDLPR-NS can be applied to recommend any kind of objects with social tag annotations, while LP-DSA specifically focus on the recommendation of location-sensitive objects based on geographical topics.

In addition, the user profile in HDLPR-NS is modelled from users' historical tagging activities, reflecting solely their long-term interests. Differently, LP-DSA considers both users' long-term interests and their real-time geographical context for person-

Table 7.1: Comparisons of the proposed recommendation models

	Application Area	Personalization	Main Challenge	Technique Used
HDLPR-NS	tag-aware	interests	uncontrolled vocabulary in social tag space	deep learning
LP-DSA	topic-based location-aware	interests & locations	sparsity, redundancy and high-dimensionality in Wikipedia topic space	deep learning
OntSim	light-weight tag-aware	interests	disambiguation without model training	ontology

alization. Consequently, LP-DSA is better at capturing the evolve of users real-time preference with the change of locations.

Since HDLPR-NS and LP-DSA have similar challenges (sparsity, ambiguity, redundancy, and high-dimensionality) in different context (social tag space and Wikipedia-based topic space), both of them are extensions of a basic recommendation-oriented deep model, deep-semantic similarity-based personalized recommendation (DSPR) model, that is presented in 4. Therefore, due to its repeated success in solving the sparsity, ambiguity, redundancy, and high-dimensionality problems, and its capability in modelling recommendation-effective topic feature representations, DSPR is a potential effective solution for similar problems in other topic-based recommendation, question answering or search applications [84, 105, 125, 129, 194, 177], where an advanced model, HDLPR-NS or LP-DSA, may be applied to further enhance the model's efficiency or bring in location-aware modelling.

In addition, since the proposed recommendation-oriented deep models have much more stable performance than the baselines, i.e., clustering and autoencoders, in the cross-domain Delicious dataset, they may also greatly contribute to the rapidly devel-

oping applications of cross-domain recommendation [49, 168] and multi-task learning [10, 77, 98] on the Social Web.

We then compare HDLRP-NS with the ontology-based recommendation model, OntSim. Similar to HDLPR-NS, OntSim is also proposed in the context of tag-aware personalized recommendation for disambiguation purpose. However, it is specifically designed for light-weight recommender systems, which alleviate the need of powerful computing facilities in deep model training.

The tag-aware recommendation performances of HDLRP-NS and OntSim have been compared in Section 6.3.3. The results show that HDLPR-NS achieve better recommendation accuracy than OntSim, with the cost of much higher computational cost. Therefore, HDLPR-NS is generally a better choice for tag-aware recommendation, while OntSim is a reasonable disambiguation alternative for light-weight recommender systems without powerful computing capability. The potential applications of OntSim on the social Web are item recommendation plugins for small-size online shops, friend recommendation services in social networking iOS/Android apps, and so on.

CHAPTER 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, I developed effective and efficient personalized recommender systems on the Social Web using social annotations, which mainly focused on the following three important sub-fields: tag-aware personalized recommender systems, location-aware personalized news recommender systems, and ontology-based personalized recommender systems.

First of all, in Chapter 4, to address the sparsity, ambiguity, and redundancy problems in the social tag space, we proposed a tag-aware personalized recommender system using a deep-semantic similarity model, DSPR, to extract dense, abstract, and recommendation-oriented representations for social tags. Experimental studies showed that DSPR achieved superior performance in tag-aware personalized recommendations; however, it had relatively high training time-cost. Consequently, we further

proposed an improved model, HDLPR-NS, to use hybrid deep learning and negative sampling to greatly reduce the system's training time and to ensure a good scalability in practice. Experiments showed that DSPR and HDLPR-NS both significantly outperformed the state-of-the-art baselines in personalized recommendation in terms of MRR, $P@k$, $R@k$, and $F@k$. In addition, the training efficiency of HDLPR-NS was hundreds of times better than the one of DSPR, while maintaining a similar training quality and performance in personalized recommendation.

Second, for the location-aware news recommendation in Chapter 5, we identified two drawbacks of the state-of-the-art location-aware news recommendation model, ELSA: (i) there exists high-dimensionality, sparsity, and redundancy problems in the Wikipedia-based topic space of ELSA, and (ii) ELSA does not use the users' long-term personal interests for personalized news recommendation. Motivated by these observations, to improve the performance of location-aware news recommender systems, we first proposed the CLSA, ALSA, and DLSA models to apply topic feature modelling to resolve the former problem; then we proposed a LP-ESA model to address the latter problem by extending ELSA to also consider the user's interests for news recommendation. Finally, we proposed a hybrid model, LP-DSA, which combined and had the advantages of DLSA and LP-ESA. Extensive experiments on a public real-world dataset showed that: (i) The proposed CLSA, ALSA, and DLSA models all greatly outperformed the state-of-the-art geographic topic model, ELSA, in location-aware news recommendations in terms of both the recommendation effectiveness and efficiency, where DLSA further outperformed CLSA and ALSA in recommendation effectiveness. (ii) LP-ESA significantly outperformed the baselines, ELSA and STPM, in terms of all evaluation metrics. (iii) The hybrid model, LP-DSA, that integrated DLSA and LP-ESA, was capable to take the advantages of both models and result in

higher recommendation effectiveness than DLSA and LP-ESA. (iv) With the help of deep neural networks, LP-DSA and DLSA had much lower online recommendation time-cost (about 25 times quicker) than LP-ESA and ELSA.

Third, in Chapter 6, we proposed ontology-based personalized recommender systems which utilize a novel ontological similarity measure to address ambiguity and redundancy problems in the social tag space without the need of model training. The ontological similarity used the concept similarity in ontologies to estimate the semantic relevance of tags in profiles, which was then used to semantically measure the similarity between user and item profiles. In Chapter 6, we first developed a two-step top-down disambiguation algorithm in order to solve the multiple occurrence challenge caused by ambiguous tags at the stage of tag allocation. Then, an algorithm was proposed to use the concept similarity in the ontology to compute the ontological similarity. Finally, an algorithm that uses the ontological similarity for content-based tag-aware personalized recommendation was proposed. A complexity analysis of these algorithms showed that the approach of using ontological similarity for content-based tag-aware personalized recommendation is practically tractable. Furthermore, extensive experimental studies based on a public real-world dataset showed that (i) the proposed ontology-based solution greatly outperformed (more than double performance) the state-of-the art baselines in tag disambiguation; (ii) the proposed top-down traversal tag allocation strategy was much more efficient than the existing tag allocation strategy which uses the Wu and Palmer similarity, while maintaining a similar tag allocation accuracy; (iii) the proposed ontological-similarity-based recommender systems were 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; OntSim is a reasonable

disambiguation alternative for light-weight recommender systems without powerful computing capability; and (iv) although HDLPR-NS generally achieves higher accuracy in tag-aware recommendation, it also has much higher computational cost; therefore, ontological similarity is a reasonable disambiguation alternative for light-weight recommender systems without powerful computing capability.

Finally, in Chapter 7, we compare the proposed recommendation models to analyze their pros and cons and also give some advice on their potential extensions and applications.

8.2 Future Work

In the future, we will further improve the proposed deep models, DSPR and HDLPR-NS, with more sophisticated neural networks to adapt to different online application scenarios. For example, in some real-world contexts, users' preferences evolve with time (e.g., one may prefer swimming in summer, but skiing in winter); as long short-term memory (LSTM) [67] networks are a kind of recurrent neural networks that are designed to better process sequential data, when the proposed DSPR or HDLPR-NS are applied to such circumstances, it will be beneficial to use LSTM networks to capture the changes of users' preferences over time.

Another future extension is to integrate the recommendation-oriented deep model, HDLPR-NS, with collaborative filtering. One possible way is to combine HDLPR-NS with matrix factorization to achieve hybrid deep matrix factorization (HDMF). By using the social tagging information as additional source of information, HDMF is expected to better model the decomposed user and item matrices in matrix factorization and remedy the cold start problem in the traditional matrix factorization solutions.

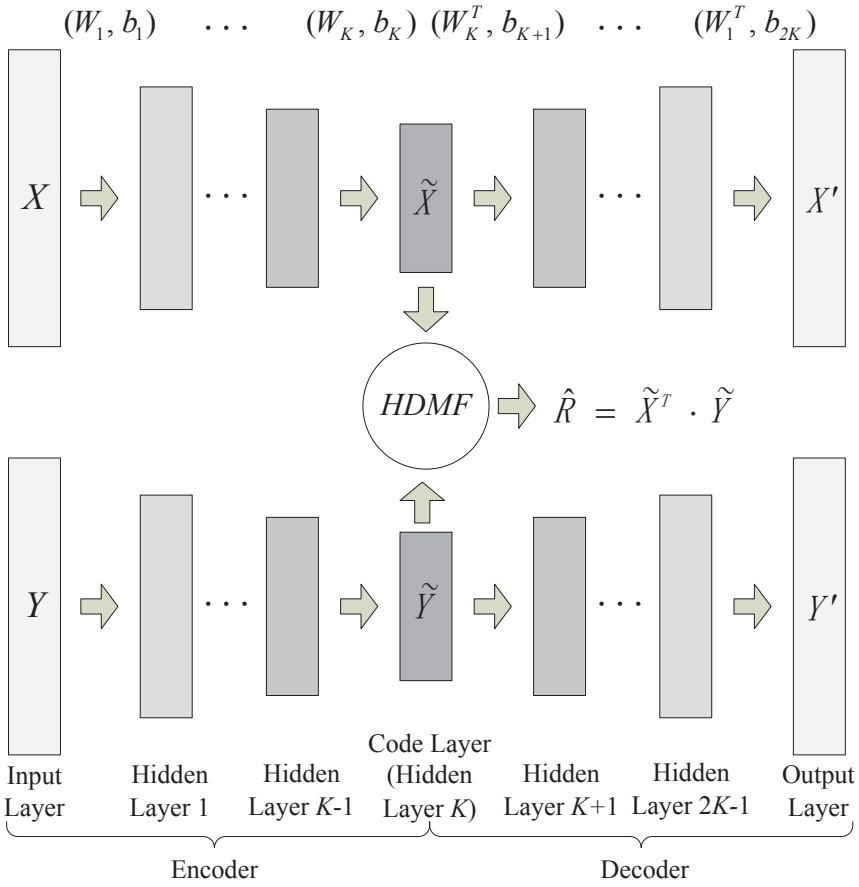


Figure 8.1: Overview of HDMF

Figure 8.1 is the expected structure of HDMF. Similar to HDLPR-NS, HDMF takes user and item profiles as inputs to model abstract feature representations of users and items ($\widetilde{X}$ and $\widetilde{X}$) in the code layers, which are used as the decomposed user and item matrices in matrix factorization. the model is thus trained to minimize the squared difference between the user-item rating matrix R and the predicted rating matrix $\hat{R} = \widetilde{X}^T \cdot \widetilde{Y}$.

Besides recommender systems, HDLPR-NS can also be applied to many potential applications in other areas. For example, it can be used to extract effective latent features from magnetic resonance imaging (MRI) pictures or x-ray images to help diagnose potential diseases. One possible way is to first replace each neural layer in

HDLPR-NS to a convolution layer and a pooling layer to extend HDLPR-NS to a convolutional hybrid deep model. Then, we represent the profile of a patient as a set of his/her historical MRI pictures, and represent the profile of a disease as a set of MRI pictures belonging to patients suffering from this disease. These profiles are used as inputs of the convolutional hybrid deep model to learn abstract feature representations of patients and disease at the code layer. Finally, by training the deep model with an objective to minimize (resp., maximize) the differences between the representations of patients and their real diseases (resp., other irrelevant diseases), the abstract features learnt by this model are expected to be effective in distinguishing different diseases, which is very helpful for diagnosis.

As for location-aware news recommendation, an interesting future work is to integrate the geographical-topic-based solutions, e.g., LP-DSA, with the physical-distance-based solutions to better recommend some location-sensitive items, e.g., activities, restaurants or tours. The intuition is that although the physical distance usually has high impact on the location-sensitive objects, the geographical topics of the candidate object sometimes will also influence the choices of users. For example, a good tour recommender system should be able to dynamically balance the distance and topic relevance of attractions. In addition, besides location information, it is also beneficial to consider more contextual information, such as timeliness of news and the social relationships of users, to achieve even better personalized recommendations.

A topic for future researches in ontology-based personalized recommendation is to conduct further experimental studies involving additional social annotation datasets, different ontologies, and other concept similarity metrics to investigate the performance change of the proposed ontological similarity under different circumstances. In addition, another topic for future research is to further consider the tag relatedness

measures (e.g., co-occurrence tags and distributional measures in [110]) as alternative solutions to address the ambiguity and redundancy problems for the tags that can not be mapped to ontologies.

BIBLIOGRAPHY

[7] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Analyzing user modeling on Twitter for personalized news recommendations. *User Modeling, Adaption and Personalization*, 1:1–12, 2011.

[8] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 147–154, 2015.

[9] Sofia Angeletou, Marta Sabou, and Enrico Motta. Semantically enriching folksonomies with FLOR. In *Proceedings of the Extended Semantic Web Conference (ESWC) Workshop on Collective Intelligence & the Semantic Web*, 2008.

[10] Trapit Bansal, David Belanger, and Andrew McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 107–114, 2016.

[11] Jie Bao, M.F. Mokbel, and Chi-Yin Chow. GeoFeed: A location aware news feed system. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*, pages 54–65, 2012.

[12] Jie Bao and Mohamed F. Mokbel. GeoRank: An efficient location-aware news feed ranking system. In *Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 184–193, 2013.

[13] Jie Bao, Yu Zheng, and Mohamed F Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings*

of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL), pages 199–208, 2012.

[14] Shenghua Bao, Guirong Xue, Xiaoyuan Wu, Yong Yu, Ben Fei, and Zhong Su. Optimizing web search using social annotations. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 501–510, 2007.

[15] Yaniv Bar, Idit Diamant, Lior Wolf, Sivan Lieberman, Eli Konen, and Hayit Greenspan. Chest pathology detection using deep learning with non-medical training. In *Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI)*, pages 294–297. IEEE, 2015.

[16] Ana Belén Barragáns-Martínez, Enrique Costa-Montenegro, Juan C Burguillo, Marta Rey-López, Fernando A Mikic-Fonte, and Ana Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences*, 180(22):4290–4311, 2010.

[17] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[18] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pages 153–160, 2007.

[19] Dominik Benz, Andreas Hotho, Robert Jäschke, Beate Krause, and Gerd Stumme. Query logs as folksonomies. *Datenbank-Spektrum*, 10(1):15–24, 2010.

[20] Daniel Billsus and Michael J Pazzani. A hybrid user model for news story classification. In *Proceedings of the International Conference on User Modeling (UM)*, pages 99–108. 1999.

[21] Daniel Billsus, Michael J. Pazzani, and James Chen. A learning agent for wireless news access. In *Proceedings of International Conference on Intelligent User Interfaces (IUI)*, pages 33–36, 2000.

[22] Kerstin Bischoff, Claudiu S. Firan, Wolfgang Nejdl, and Raluca Paiu. Can all tags be used for search? In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 193–202, 2008.

[23] S. Bjornestad, B. Tessem, and L. Nyre. Design and evaluation of a location-based mobile news reader. In *Proceedings of IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–4, 2011.

[24] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[25] Jesus Bobadilla, Antonio Hernando, Fernando Ortega, and Jesus Bernal. A framework for collaborative filtering recommender systems. *Expert Systems with Applications*, 38(12):14609–14623, 2011.

[26] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[27] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-Based System*, 46:109–132, 2013.

[28] JESUS Bobadilla, Francisco Serradilla, Antonio Hernando, et al. Collaborative filtering adapted to recommender systems of e-learning. *Knowledge-Based Systems*, 22(4):261–265, 2009.

[29] Léon Bottou. Stochastic learning. In *Advanced lectures on machine learning*, pages 146–168. 2004.

[30] Mohamed Reda Bouadjenek, Hakim Hacid, and Mokrane Bouzeghoub. Sopra: a new social personalized ranking function for improving web search. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 861–864, 2013.

[31] Mohamed Reda Bouadjenek, Hakim Hacid, and Mokrane Bouzeghoub. Social networks and information retrieval, how are they converging? a survey, a taxonomy and an analysis of social information retrieval approaches and platforms. *Information Systems*, 56:1–18, 2016.

[32] Mohamed Reda Bouadjenek, Hakim Hacid, Mokrane Bouzeghoub, and Athena Vakali. Using social annotations to enhance document representation for personalized search. In *Proceedings of the international ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1049–1052, 2013.

[33] Iván Cantador, Alejandro Bellogín, and David Vallet. Content-based recoemmndation in social tagging systems. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 237–240, 2010.

[34] Ivan Cantador, Peter Brusilovsky, and Tsvi Kuflik. Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011). In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 387–388, 2011.

[35] Andrea Capocci and Guido Caldarelli. Folksonomies and clustering in the collaborative system citeulike. *Journal of Physics A: Mathematical and Theoretical*, 41(22):224016, 2008.

[36] Walter Carrer-Neto, María Luisa Hernández-Alcaraz, Rafael Valencia-García, and Francisco García-Sánchez. Social knowledge-based recommender system. application to the movies domain. *Expert Systems with applications*, 39(12):10990–11000, 2012.

[37] Jose Jesus Castro-Schez, Raul Miguel, David Vallejo, and Lorenzo Manuel López-López. A highly adaptive recommender system based on fuzzy logic for b2c e-commerce portals. *Expert Systems with Applications*, 38(3):2441–2454, 2011.

[38] Ciro Cattuto, Dominik Benz, Andreas Hotho, and Gerd Stumme. Semantic grounding of tag relatedness in social bookmarking systems. In *Proceedings of International Semantic Web Conference (ISWC)*, pages 615–631, 2008.

[39] Cen Chen, Peilin Zhao, Longfei Li, Jun Zhou, Xiaolong Li, and Minghui Qiu. Locally connected deep learning framework for industrial-scale recommender systems. In *Proceedings of the International Conference on World Wide Web (WWW) Companion*, pages 769–770, 2017.

[40] Chih-Ming Chen. An intelligent mobile location-aware book recommendation system that enhances problem-based learning in libraries. *Interactive Learning Environments*, 21(5):469–495, 2013.

[41] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the Workshop on Deep Learning for Recommender Systems (DLRS)*, pages 7–10, 2016.

[42] Zhiyong Cheng, Shen Jialie, and Steven CH Hoi. On effective personalized music retrieval by exploring online user behaviors. In *Proceedings of the International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR)*, pages 125–134, 2016.

[43] Paulo Chiliguano and Gyorgy Fazekas. Hybrid music recommender using content-based and social information. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2618–2622, 2016.

[44] Wei Chu and Seung-Taek Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 691–700, 2009.

[45] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for YouTube recommendations. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 191–198, 2016.

[46] Rubén González Crespo, Oscar Sanjuán Martínez, Juan Manuel Cueva Lovelle, B Cristina Pelayo García-Bustelo, José Emilio Labra Gayo, and Patricia Ordonez De Pablos. Recommendation system based on user interaction data applied to intelligent electronic books. *Computers in Human Behavior*, 27(4):1445–1449, 2011.

[47] Maria-Iuliana Dascalu, Constanta-Nicoleta Bodea, Alin Moldoveanu, Anca Mohora, Miltiadis Lytras, and Patricia Ordoñez de Pablos. A recommender agent based on learning styles for better virtual collaborative learning experiences. *Computers in Human Behavior*, 45:243–253, 2015.

[48] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: A search and metadata engine for the semantic web. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 652–659, 2004.

[49] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 278–288, 2015.

[50] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

[51] Evgeniy Gabrilovich and Shaul Markovitch. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, 34(1):443–498, 2009.

[52] Prasanna Ganesan, Hector Garcia-Molina, and Jennifer Widom. Exploiting hierarchical domain structure to compute similarity. *ACM Transaction on Information System*, 21(1):64–93, 2003.

[53] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. Mobile recommender systems in tourism. *Journal of network and computer applications*, 39:319–333, 2014.

[54] Fatih Gedikli and Dietmar Jannach. Rating items by rating tags. In *Proceedings of ACM Conference on Recommender Systems (RecSys)*, pages 25–32, 2010.

[55] Jonathan Gemmell, Thomas Schimoler, M Ramezani, L Christiansen, and B Mobasher. Resource recommendation for social tagging: a multi-channel hybrid approach. In *Proceedings of ACM Conference on Recommender Systems (RecSys)*, pages 60–67, 2010.

[56] Jonathan Gemmell, Thomas Schimoler, Maryam Ramezani, Laura Christiansen, and Bamshad Mobasher. Improving FolkRank with item-based collaborative filtering. In *Proceedings of ACM Conference on Recommender Systems (RecSys)*, pages 17–24, 2009.

[57] Yuyun Gong and Qi Zhang. Hashtag recommendation using attention-based convolutional neural network. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2782–2788, 2016.

[58] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies - Special issue: the role of formal ontology in the information technology*, 43(5-6):907–928, 1995.

[59] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction. In *arXiv preprint arXiv:1703.04247*, 2017.

[60] Mohamed Soliman Halawa, Essam M Ramzy Hamed, and Mohamed Elemam Shehab. Personalized e-learning recommendation model based on psychological type and learning style models. In *Proceedings of the International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 578–584, 2015.

[61] Xiaogang Han, Zhiqi Shen, Chunyan Miao, and Xudong Luo. Folksonomy-based ontological user interest profile modeling and its application in personalized search. In *Proceedings of the International Conference on Active Media Technology (AMT)*, pages 34–46, 2010.

[62] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 173–182, 2017.

[63] Anvitha Hegde and Savitha K Shetty. Collaborative filtering recommender system. *International Journal of Emerging Trends in Science and Technology*, 2(07), 2015.

[64] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[65] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[66] Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In *WordNet: An Electronic Lexical Database*, pages 305–332, 1998.

[67] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[68] T Ryan Hoens, Marina Blanton, Aaron Steele, and Nitesh V Chawla. Reliable medical recommendation systems with patient privacy. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(4):67, 2013.

[69] Frank Hopfgartner, Torben Brodt, Jonas Seiler, Benjamin Kille, Andreas Lommatzsch, Martha Larson, Roberto Turrin, and András Serény. Benchmarking news recommendations: The clef newsreel use case. In *ACM SIGIR Forum*, volume 49, pages 129–136. ACM, 2016.

[70] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 411–426, 2006.

[71] I-Ching Hsu. Integrating ontology technology with folksonomies for personalized social tag recommendation. *Applied Soft Computing*, 13(8):3745–3750, 2013.

[72] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2333–2338, 2013.

[73] Wenyi Huang, Zhaohui Wu, Chen Liang, Prasenjit Mitra, and C Lee Giles. A neural probabilistic model for context based citation recommendation. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 2404–2410, 2015.

[74] Zan Huang, Daniel Zeng, and Hsinchun Chen. A comparison of collaborative-filtering recommendation algorithms for e-commerce. *IEEE Intelligent Systems*, 22(5), 2007.

[75] Zhengxing Huang, Xudong Lu, Huilong Duan, and Chenhui Zhao. Collaboration-based medical knowledge recommendation. *Artificial intelligence in Medicine*, 55(1):13–24, 2012.

[76] Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[77] How Jing and Alexander J Smola. Neural survival recommender. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 515–524, 2017.

[78] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 154–161, 2005.

[79] Alexandros Karatzoglou, Balázs Hidasi, Domonkos Tikk, Oren Sar-Shalom, Haggai Roitman, and Bracha Shapira. Recsys' 16 workshop on deep learning for recommender systems (dlrs). In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 415–416, 2016.

[80] Gabriella Kazai, Iskander Yusof, and Daoud Clarke. Personalised news and blog recommendations based on user location, Facebook and Twitter user profiling. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1129–1132, 2016.

[81] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 233–240, 2016.

[82] Aleksandra Klašnja-Milićević, Mirjana Ivanović, and Alexandros Nanopoulos. Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review*, 44(4):571–604, 2015.

[83] Beate Krause, Andreas Hotho, and Gerd Stumme. A comparison of social bookmarking with traditional search. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, pages 101–113, 2008.

[84] Takeshi Kurashima, Tomoharu Iwata, Go Irie, and Ko Fujimura. Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 579–588, 2010.

[85] Nikolas Landia and S Anand. Personalised tag recommendation. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 83–86, 2009.

[86] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 10, pages 331–339, 1995.

[87] Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 467–476, 2009.

[88] Seok Kee Lee, Yoon Ho Cho, and Soung Hie Kim. Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations. *Information Sciences*, 180(11):2142–2155, 2010.

[89] Wei-Po Lee and Cheng-Che Lu. Customising WAP-based information services on mobile networks. *Personal and Ubiquitous Computing*, 7(6):321–330, 2003.

[90] Chenyi Lei, Dong Liu, Weiping Li, Zheng-Jun Zha, and Houqiang Li. Comparative deep learning of hybrid representations for image recommendations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2545–2553, 2016.

[91] Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. LARS: A location-aware recommender system. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 450–461. IEEE, 2012.

[92] Chenliang Li, Aixin Sun, and Anwitaman Datta. A generalized method for word sense disambiguation based on Wikipedia. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, pages 653–664, 2011.

[93] Chenliang Li, Aixin Sun, and Anwitaman Datta. Tsdw: Two-stage word sense disambiguation using Wikipedia. *Journal of the American Society for Information Science and Technology*, 64(6):1203–1223, 2013.

[94] Huayu Li, Hengshu Zhu, Yong Ge, Yanjie Fu, and Yuan Ge. Personalized tv recommendation with mixture probabilistic matrix factorization. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 352–360, 2015.

[95] Hui Li, Jiangtao Cui, Bingqing Shen, and Jianfeng Ma. An intelligent movie recommendation system through group-level sentiment analysis in microblogs. *Neurocomputing*, 210:164–173, 2016.

[96] Kai Li and Timon C Du. Building a targeted mobile advertising system for location-based services. *Decision Support Systems*, 54(1):1–8, 2012.

[97] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 661–670, 2010.

[98] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. Neural rating regression with abstractive tips generation for recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2017.

[99] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. Mining user similarity based on location history. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, page 34, 2008.

[100] Seth Siyuan Li and Elena Karahanna. Online recommendation systems in a b2c e-commerce context: a review and future directions. *Journal of the Association for Information Systems*, 16(2):72, 2015.

[101] Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 811–820, 2015.

[102] Yuhua Li, Zuhair A. Bandar, and David McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882, 2003.

[103] Jianxun Lian, Fuzheng Zhang, Xing Xie, and Guangzhong Sun. Cccfnet: A content-boosted collaborative filtering neural network for cross domain recommender systems. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 817–818, 2017.

[104] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 296–304, 1998.

[105] Bin Liu and Hui Xiong. Point-of-interest recommendation in location based social networks with topic and location awareness. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 396–404, 2013.

[106] Yao Lu, Zhi Qiao, Chuan Zhou, Yue Hu, and Li Guo. Location-aware friend recommendation in event-based social networks: A Bayesian latent factor approach. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1957–1960, 2016.

[107] Zhongqi Lu, Zhicheng Dou, Jianxun Lian, Xing Xie, and Qiang Yang. Content-based collaborative filtering for news topic recommendation. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 217–223, 2015.

[108] Xin Luo, Yunni Xia, and Qingsheng Zhu. Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems*, 27:271–280, 2012.

[109] Julia Luxenburger, Shady Elbassuoni, and Gerhard Weikum. Task-aware search personalization. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 721–722, 2008.

[110] Benjamin Markines, Ciro Cattuto, Filippo Menczer, Dominik Benz, Andreas Hotho, and Gerd Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 641–650, 2009.

[111] Alessandro Micarelli and Filippo Sciarrone. Anatomy and empirical evaluation of an adaptive Web-based information filtering system. *User Modeling and User-Adapted Interaction*, 14(2-3):159–200, 2004.

[112] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[113] David Milne and Ian H. Witten. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2008.

[114] Hamed Movahedian and Mohammad Reza Khayyambashi. Folksonomy-based user interest and disinterest profiling for improved recommendations: An ontological approach. *Journal of Information Science*, 40(5):594–610, 2014.

[115] Alexandros Nanopoulos, Dimitrios Rafailidis, Panagiotis Symeonidis, and Yannis Manolopoulos. Musicbox: Personalized music recommendation based on cubic analysis of social tags. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(2):407–412, 2010.

[116] Yunseok Noh, Yong-Hwan Oh, and Seong-Bae Park. A location-based personalized news recommendation. In *Proceedings of International Conference on Big Data and Smart Computing (BIGCOMP)*, pages 99–104, 2014.

[117] Edward Rolando Núñez-Valdéz, Juan Manuel Cueva Lovelle, Oscar Sanjuán Martínez, Vicente García-Díaz, Patricia Ordoñez De Pablos, and Carlos Enrique Montenegro Marín. Implicit feedback techniques on recommender systems applied to electronic books. *Computers in Human Behavior*, 28(4):1186–1193, 2012.

[118] Kenta Oku, Shinsuke Nakajima, Jun Miyazaki, and Shunsuke Uemura. Context-aware svm for context-dependent information recommendation. In *Proceedings of the International Conference on Mobile Data Management (MDM)*, pages 109–109, 2006.

[119] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. Embedding-based news recommendation for millions of users. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1933–1942, 2017.

[120] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the Web. *Stanford InfoLab*, 1999.

[121] Moon-Hee Park, Jin-Hyuk Hong, and Sung-Bae Cho. Location-based recommendation system using bayesian users preference model in mobile devices. In *Proceedings of the International Conference on Ubiquitous Intelligence and Computing (UIC*, pages 1130–1139, 2007.

[122] Michael Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill & Webert: Identifying interesting web sites. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 54–61, 1996.

[123] Chao-Ying Joanne Peng, Kuk Lida Lee, and Gary M Ingersoll. An introduction to logistic regression analysis and reporting. *The Journal of Educational Research*, 96(1):3–14, 2002.

[124] Euripides G. M. Petrakis, Giannis Varelas, Angelos Hliaoutakis, and Paraskevi Raftopoulou. X-Similarity: Computing semantic similarity between concepts from different ontologies. *Journal of Digital Information Management*, 4, 2006.

[125] Konstantinos Pliakos and Constantine Kotropoulos. PLSA driven image annotation, classification, and tourism recommendation. In *IEEE International Conference on Image Processing (ICIP)*, pages 3003–3007, 2014.

[126] Carlos Porcel and Enrique Herrera-Viedma. Dealing with incomplete information in a fuzzy linguistic recommender system to disseminate information in university digital libraries. *Knowledge-Based Systems*, 23(1):32–39, 2010.

[127] Carlos Porcel, A Tejeda-Lorente, MA Martínez, and Enrique Herrera-Viedma. A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. *Information Sciences*, 184(1):1–19, 2012.

[128] Silvia Puglisi, Javier Parra-Arnau, Jorge Forné Muñoz, and David Rebollo Monedero. On content-based recommendation and user privacy in social-tagging systems. *Computer Standards & Interfaces*, 41:17–27, 2015.

[129] Shinjee Pyo, Eunhui Kim, et al. LDA-based unified topic modeling for similar TV user grouping and TV program recommendation. *IEEE transactions on cybernetics*, 45(8):1476–1490, 2015.

[130] Zhi Qiao, Peng Zhang, Yanan Cao, Chuan Zhou, Li Guo, and Binxing Fang. Combining heterogenous social and geographical information for event recommendation. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, volume 14, pages 145–151, 2014.

[131] Kjetil Vaage Qie. Sensing the news: User experiences when reading locative news. *Future Internet*, 4:161–178, 2012.

[132] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30, 1989.

[133] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1375–1384, 2011.

[134] Hadas Raviv, Oren Kurland, and David Carmel. Document retrieval using entity-based language models. In *Proceedings of the International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR)*, pages 65–74, 2016.

[135] Pedro M. P. Rosa, Joel J. P. C. Rodrigues, and Filippo Basso. A weight-aware recommendation algorithm for mobile multimedia systems. *Mobile Information Systems*, 9(2):139–155, 2013.

[136] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 791–798, 2007.

[137] James Salter and Nick Antonopoulos. Cinemascreen recommender agent: combining collaborative and content-based filtering. *IEEE Intelligent Systems*, 21(1):35–41, 2006.

[138] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[139] Mohamed Sarwat, Jie Bao, Ahmed Eldawy, Justin J. Levandoski, Amr Magdy, and Mohamed F. Mokbel. Sindbad: a location-based social networking system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 649–652. ACM, 2012.

[140] Mohamed Sarwat, Justin J. Levandoski, Ahmed Eldawy, and Mohamed F. Mokbel. LARS*: An efficient and scalable location-aware recommender system. *IEEE Transactions on Knowledge and Data Engineering*, 26(6):1384–1399, 2014.

[141] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas. Music recommender systems. In *Recommender Systems Handbook*, pages 453–492. 2015.

[142] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 111–112, 2015.

[143] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Representation learning of users and items for review rating prediction using attention-based convolutional neural network. In *Proceedings of the International Workshop on Machine Learning Methods for Recommender Systems (MLRec)*, 2017.

[144] Jesus Serrano-Guerrero, Enrique Herrera-Viedma, Jose A Olivas, Andres Cerezo, and Francisco P Romero. A Google wave-based fuzzy recommender system to disseminate information in university digital libraries 2.0. *Information Sciences*, 181(9):1503–1516, 2011.

[145] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI)*, pages 210–217, 1995.

[146] X. Shen and C. X. Zhai. Exploiting query history for document ranking in interactive information retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 377–378, 2003.

[147] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 824–831, 2005.

[148] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 259–266, 2008.

[149] Fabrizio Silvestri. Mining query logs: Turning search usage data into knowledge. *Foundations and Trends in Information Retrieva*, 4:1–174, 2010.

[150] Márcio Soares and Paula Viana. Tuning metadata for better movie content-based recommendation systems. *Multimedia Tools and Applications*, 74(17):7015–7036, 2015.

[151] Shahab Saquib Sohail, Jamshed Siddiqui, and Rashid Ali. OWA based book recommendation technique. *Procedia Computer Science*, 62:126–133, 2015.

[152] Jeong-Woo Son, A-Yeong Kim, and Seong-Bae Park. A location-based news article recommendation with explicit localized semantic analysis. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 293–302, 2013.

[153] Jeong-Woo Son, Yun-Seok Noh, Hyun-Je Song, and Seong-Bae Park. Location comparison through geographical topics. In *Proceedings of IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology (WI/IAT)*, pages 311–318, 2012.

[154] Mirco Speretta. Personalized search based on user search histories. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 622–628, 2005.

[155] Florian Strub, Romaric Gaudel, and Jérémie Mary. Hybrid recommender system based on autoencoders. In *Proceedings of the Workshop on Deep Learning for Recommender Systems (DLRS)*, pages 11–16, 2016.

[156] Florian Strub and Jérémie Mary. Collaborative filtering with stacked denoising autoencoders and sparse inputs. In *Proceedings of the NIPS workshop on Machine Learning for eCommerce*, 2015.

[157] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. Adaptive Web search based on user profile constructed without any effort from users. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 675–684, 2004.

[158] Shulong Tan, Jiajun Bu, Chun Chen, and Xiaofei He. Using rich social media information for music recommendation via hypergraph model. In *Social media modeling and computing*, pages 213–237. 2011.

[159] Jaime Teevan, Susan T. Dumais, and Daniel J. Liebling. To personalize or not to personalize: modeling queries with variation in user intent. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 163–170, 2008.

[160] Karen HL Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pages 1995–1999, 2008.

[161] Jeffrey D Ullman, Jure Leskovec, and Anand Rajaraman. Mining of massive datasets, 2011.

[162] Moshe Unger. Latent context-aware recommender systems. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 383–386, 2015.

[163] Moshe Unger, Ariel Bar, Bracha Shapira, and Lior Rokach. Towards latent context-aware recommendation systems. *Knowledge-Based Systems*, 104:165–178, 2016.

[164] David Vallet, Ivn Cantador, and Joemon M. Jose. Personalizing web search with folksonomy-based user and document profiles. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, pages 420–431, 2010.

[165] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pages 2643–2651, 2013.

[166] Ellen M Voorhees. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82, 1999.

[167] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. What your images reveal: Exploiting visual contents for point-of-interest recommendation. In *Proceedings of the 26th International World Wide Web Conference (WWW)*, pages 391–400, 2017.

[168] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item Silk Road: Recommending items from information domains to social users. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2017.

[169] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the ACM international conference on Multimedia (ACM-MM)*, pages 627–636, 2014.

[170] Pinata Winoto and Tiffany Y Tang. The role of user mood in movie recommendations. *Expert Systems with Applications*, 37(8):6086–6092, 2010.

[171] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, and Alexander J Smola. Joint training of ratings and reviews with recurrent recommender networks. 2017.

[172] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 495–503, 2017.

[173] Sai Wu, Weichao Ren, Chengchao Yu, Gang Chen, Dongxiang Zhang, and Jingbo Zhu. Personal recommendation using deep recurrent neural networks in netease. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 1218–1229, 2016.

[174] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 153–162, 2016.

[175] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 133–138, 1994.

[176] Ruobing Xie, Zhiyuan Liu, Rui Yan, and Maosong Sun. Neural emoji recommendation in dialogue systems. *arXiv preprint arXiv:1612.04609*, 2016.

[177] Fei Xu, Zongcheng Ji, and Bin Wang. Dual role model for question recommendation in community question answering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 771–780, 2012.

[178] Shengliang Xu, Shenghua Bao, Ben Fei, Zhong Su, and Yong Yu. Exploring folksonomy for personalized search. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 155–162, 2008.

[179] Wenjian Xu, Chi-Yin Chow, Man Lung Yiu, Qing Li, and Chung Keung Poon. MobiFeed: a location-aware news feed system for mobile users. In *Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 538–541, 2012.

[180] Wenjian Xu, Chi-Yin Chow, Man Lung Yiu, Qing Li, and Chung Keung Poon. MobiFeed: a location-aware news feed framework for moving users. *GeoInformatica*, 19(3):633–669, 2015.

[181] Carl Yang, Lanxiao Bai, Chao Zhang, and Jiawei Han. Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 353–362, 2017.

[182] Wan-Shiou Yang, Hung-Chi Cheng, and Jia-Ben Dia. A location-aware recommender system for mobile shopping environments. *Expert Systems with Applications*, 34(1):437–445, 2008.

[183] Lina Yao, Quan Z Sheng, Anne HH Ngu, Jian Yu, and Aviv Segev. Unified collaborative and content-based web service recommendation. *IEEE Transactions on Services Computing*, 8(3):453–466, 2015.

[184] Kam Fung Yeung and Yanyan Yang. A proactive personalized mobile news recommendation system. In *Proceedings of Developments in E-Systems Engineering (DESE)*, pages 207–212, 2010.

[185] Hongzhi Yin, Bin Cui, Yizhou Sun, Zhiting Hu, and Ling Chen. LCARS: a spatial item recommender system. *ACM Transactions on Information Systems*, 32(3):1–37, 2014.

[186] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. LCARS: a location-content-aware recommender system. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 221–229, 2013.

[187] Xiao Yingyuan, Ai Pengqiang, Hsu Ching-Hsien, Wang Hongya, and Jiao Xu. Time-ordered collaborative filtering for news recommendation. *China Communications*, 12(12):53–62, 2015.

[188] Chien-Chih Yu and Hsiao-Ping Chang. Personalized location-based recommendation services for tour planning in mobile tourism applications. *E-Commerce and Web Technologies*, pages 38–49, 2009.

[189] Zhiwen Yu, Xingshe Zhou, Yanbin Hao, and Jianhua Gu. TV program recommendation for multiple viewers based on user profile merging. *User modeling and user-adapted interaction*, 16(1):63–82, 2006.

[190] Jing Yuan, Yu Zheng, Liuhang Zhang, XIng Xie, and Guangzhong Sun. Where to find my next passenger. In *Proceedings of the International Conference on Ubiquitous Computing (Ubicomp)*, pages 109–118, 2011.

[191] Quan Yuan, Gao Cong, Kaiqi Zhao, Zongyang Ma, and Aixin Sun. Who, where, when, and what: A nonparametric bayesian approach to context-aware recommendation and search for Twitter users. *ACM Transactions on Information Systems*, 33(1):2, 2015.

[192] Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*, 2017.

[193] Yin Zhang, Min Chen, Dijiang Huang, Di Wu, and Yong Li. iDoctor: personalized and professionalized medical recommendations based on hybrid matrix factorization. *Future Generation Computer Systems*, 66:30–35, 2017.

[194] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. Auralist: introducing serendipity into music recommendation. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 13–22, 2012.

[195] Zi-Ke Zhang, Tao Zhou, and Yi-Cheng Zhang. Tag-aware recommender systems: a state-of-the-art survey. *Journal of Computer Science and Technology*, 26(5):767–777, 2011.

[196] Lili Zhao, Zhongqi Lu, Sinno Jialin Pan, and Qiang Yang. Matrix factorization+ for movie recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3945–3951, 2016.

[197] Yi Zhen, Wu-Jun Li, and Dit-Yan Yeung. Tagicofi: tag informed collaborative filtering. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 69–76, 2009.

[198] Lei Zheng. A survey and critique of deep learning on recommender systems. 2016.

[199] Lei Zheng, Vahid Noroozi, and Philip S Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 425–434, 2017.

[200] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative location and activity recommendations with GPS history data. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 1029–1038, 2010.

[201] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *ACM Transactions on the Web*, 5(1):5, 2011.

[202] Youjie Zhou and Jiebo Luo. Geo-location inference on news articles via multimodal pLSA. In *Proceedings of ACM International Conference on Multimedia (ACM-MM)*, pages 741–744, 2012.

[203] Yi Zuo, Jiulin Zeng, Maoguo Gong, and Licheng Jiao. Tag-aware recommender systems based on deep neural networks. *Neurocomputing*, 204:51–60, 2016.