ISP/PhD Comprehensive Examination

Recommender Systems: Dr. Daqing He
Educational Data Mining: Dr. Peter Brusilovsky
Machine Learning: Dr. Gregory Cooper

Shaghayegh Sahebi
Intelligent Systems Program,
University of Pittsburgh
shs106@pitt.edu

December 21, 2013

1 Recommender Systems

Konstan, Joseph A., and John Riedl, in their 2012 User Modeling and User Adapted Interaction article "Recommender systems: from algorithms to user experience" identify better exploitation of **user-contributed content**, as one of the major challenges that the field of recommender systems have to address in order to keep the field moving forward.

1.1 Review the literature on recommendation systems, analyze and summarize in your own words the existing project/studies/algorithms on recommendations *based on user contributed contents*

.

As suggested by Konstan and Riedl [66], user-contributed content is one of the vast and important data resources in recommender systems. It includes users' tags, reviews, playlists, blog posts and micro-blogging content, forums, etc. Although the aforementioned list include the mostly agreed upon forms of user-contributed content, an extreme view can even consider ratings of users or the trust links between them as a form of user-contributed information. Although link structure can be extracted from some of the user-contributed content (either explicit such as following in Twitter or implicit such as the structure extracted from retweets), I focus on the more recognized forms of user-contributed content.

To categorize user-contributed content in recommender systems, we have multiple options: based on the type of user-contributed content, the type of recommendation algorithm they are used in, the ways it is incorporated in recommender systems, and the items being recommended. Since the main focus of the question is on user-contributed contents, I choose the type of content as my main categorization factor. However, I briefly overview the other factors in the following.

As for type of recommendation algorithm that user-contributed content are used in, it can be utilized in content-based [45], collaborative filtering [79], knowledge-based [16], and hybrid recommender systems [5]. Since recommender systems in general have the same categorization, I skip introducing these recommender systems and note the type of recommender system when referring to the literature.

There are multiple ways user-contributed content blends into recommender systems. User-contributed data can be used to enrich user profiles [79, 13]. Usually, in collaborative filtering (CF), user is modeled based on the items she has rated or consumed. In the original form of CF, the items are represented only by their IDs and no other information. In real world, each user might have multiple topics of interests that each of these items might cover a part of it. One way of capturing these topics is looking at user-contributed content. By representing user profiles in the space of textual features for items, we can obtain an enriched user profile that directly shows user interests. Having the user-contributed data, we do not need to look at the items to find user interests: we just need to extract them from the information user has provided. For example, if we are on Flickr looking for a specific user's interests, we can look at the set of tags she has used for all of the photos in her profile and build her user-profile in the tag-space instead of photo-space. We do not need to look at which photos she has tagged to do this. This idea can be extended to blog or micro-blogging contents, when the users directly talk about what they like and dislike: e.g., "I prefer comedy movies to drama.". The same applies to item profiles [118]. Items can be represented in the user-contributed content's space. For example, an item can be represented by the tags it is annotated with. This provides a common medium for both users and items and makes them directly comparable to each other. Some of the recommender algorithms directly model this common medium using tensors or hypergraphs to represent the relationship between users, items, and user-contributed content [57, 97]. In other models, user-contributed content is designed as a part of the recommender system model [56, 58].

Regarding the items that get recommended using these algorithms, I categorize them into two types of internal and external recommendations. By internal recommendation, I refer to the recommender systems that aim to recommend the same user-contributed content or items in the same domain of user-contributed content to users [97, 132]. For example, if a recommender system recommends tweets based on tweet contents of a user or recommends tags for photos shared by users, it is an internal recommendation. On the other hand, if a recommender system recommends items outside the domain of user-contributed content [106, 129, 108], such as recommending people based on blog contents [122], it is an external recommendation. It should be noted that this is my categorization factor to facilitate the explanations of how user-contributed content is used in recommender systems. This definition is not used in other literature.

In the following, I present a detailed categorization of using user-contributed content in recommender systems based on the type of the content. User-contributed content can be classified as below:

- *Textual Data*
  - *User-provided Tags and Folksonomies*
  - *Weblogs and User Comments*
  - *Forums and Q&A Websites*
  - *Micro-blogging (e.g., Twitter or Facebook)*
  - *User Comments and reviews (on specific items)*
- *User-Generated Lists (e.g., playlists)*

- *User-Generated Multimedia (e.g., photos and videos)*
- *User's Geographical Data (e.g., check-ins)*

1.1.1 Textual Data

Textual data is the most common user-contributed content. By the rise of social web, the opportunities to provide content by users has been increased: users can tag almost anything from scientific papers, to photos, people, web pages, etc; users can write blog posts; comment on other user-provided or system-provided objects; and provide real-time updates in the form of micro-blogging. This rich set of information resources can be used as enrichment of user and item profiles, as a bridge to link objects and users, as an external information to build an ontology or dictionary to categorize the objects, or to understand user's opinion on a specific object. The methods of using this information can be related to a specific user (e.g., using a user's tags on photos to recommended more photos to that specific user) or as a general source of information (e.g., using review to enrich item profiles). In the following, I present some of the recommender systems that use textual user-contributed content categorized by the type of the textual content.

Tags and Folksonomies:
Folksonomy is a taxonomy generated by users who collaboratively annotate and categorize resources of interests with freely chosen keywords called tags [102]. For representation, tags can be modeled as a part of a tensor or hypergraph that shows the relationship between users, items, and tags. Tags are usually collected via social tagging, experts, annotation games, or content-based tagging (generating tags based on data mining algorithms from text/other resources which is not user-contributed). Tags have been used in many recommender systems to recommend items, users, tags, etc. Because of the extensive volume of related work in this category, I only present a handful of major tag-based recommender systems. Milicevic et al. [85] provide a more extensive survey of tagging in recommender systems.

Tags can be used to enrich user profiles by modeling the user in the tag space and representing her by the most important/representative tags for her. In the same way, they can be used to extend the item profile. This extended profile can either be used in a content-based recommender system (e.g., [118]), or in a hybrid recommender system (using collaborative filtering on the extended profiles, using a variation of item-based collaborative filtering, or moving from content-based recommenders to a content-collaborative recommender system - e.g., [79, 13]). Liang et al. [79] use user tags to extend user profiles and calculate three different tag-based similarities between users to perform collaborative filtering. Tso-Sutter et al. [118] use tags to extend both user and item profiles. Diederich et al. [29] and Stoyanovich et al. [106] use variations of weighted vector of tags for modeling user profile. They believe that each tag might have a different importance weight for each user. Yeung et al. [129] use clustering of tagged bookmarks to build multiple-interest profiles for users. They propose this model to capture the fact that each user might have more than one topic of interest. Szomszor et al. [108] use tags from IMDB in combination of usage data from Netflix to recommend movies to users. Firan et al. [34] use tag-based profiles to recommend playlists to the users. Godoy and Amandi [45] use tags to enrich the content-based user profiles. Parra and Brusilovsky [94] use CiteULike tags to extend collaborative filtering recommendations for recommending scientific articles.

In addition to user-profile extension, tags have been modeled in more complicated recommender systems that model the tags as an embedded part of the recommender algorithm. For example, Zhao et al. [133] use youtube tags with social links from Facebook and other sources of information in a graphical model to recommend movies to users. Wetzker et al. [122] proposed an extension of PLSA to jointly model the tags, items, and users for item recommendation.

Tags have been used as a common medium between users and items. In this case, mostly they are modeled as a part of a hypergraph (an undirected, weighted tripartite graph consisting of tags, items, and users) or in a tensor. The FolkRank algorithm [57] is an adaption of PageRank that uses a transformation of the hypergraph formed by the traditional tag assignments. This algorithm produces a set of related users and resources for each tag. Using the results of Folkrank, one can recommend items or tags to users. Symeonidis et al. [107] and and Rendle et al. [97] used tensor factorization on Bibsonomy and Last.fm data for tag recommendation. Song et al. [105] proposed a tag recommender based on graph clustering for tagging textual resources, such as web pages.

In addition to explaining the object that is tagged, tags can be used as a kind of feedback to infer user opinion [37]. Users use tags to provide their review on a subject matter such as "interesting", "good", "bad", etc.

As for many other user-contributed content, using tags has its own problems including ambiguity, polysemy, synonymy, trusting the user providing the tag, etc. There have been some approaches to take the noise out of tagging datasets. For example, Cantador et al. [14] proposed a methodology to select "meaningful" tags using WORDNET, Wikipedia and Google search engine.

Weblogs and User Comments:
Blogs are a medium for Web users to present their opinions. They can be professional (scientific, photo-blog, company-related, etc.), personal, or related to a community. On one hand, blog posts can be a good resource of information for finding the blogger and commenter's opinion on a subject matter and the relationship between bloggers and people who comment. On the other hand, having so many textual information in the form of blogs can be misleading and overwhelming. As a result, link structure and contents of blogs can be used both for recommending other blog posts [52] as well as items [5], tags [53], and people [16]. Avesani et al. [5] collect user experiences on ski mountaineering and other users with a blog oriented architecture (Moleskiing.it) and produce trust-aware recommendations of ski routes. Here, both blog structure and content is used to produce hybrid recommendations. Hayes et al. [53] use clustering on blog data to recommend tags to users. In this work, weblog content is used to extend user profiles. In another work [52], they use a similar idea to recommend posts similar to a blogger's current interests (content-based filtering). Caragea et al. [16] use blog contents of LiveJournal to build an ontology and then, using both this ontology and blog link structure, recommend friendship links to bloggers. Hsu et al. [59] present a link recommendation method based on blog contents and the link-structure between them. Jian et al. [61] index weblog contents and build an extended user profile using the clusters of blog content. They recommend high-quality blogs using collaborative filtering on extended user profile.

As we can see, blog content and structure is used to enhance both content-based [52] and CF [61] recommender systems towards hybrid recommendations. They are used in trust-aware recommenders [5] and knowledge-based recommender systems [16]. In addition to the hybrid models that are based on content-based and collaborative filtering, blog content can also be modeled as an integrated part of the recommender model. Cohn and Hofmann [56], introduce a joint probabilistic model to model both the content and links between the documents. This model can be used for blog data in recommender systems.

Forums and Q&A Websites:
Forums and Q&A Websites are other forms of user-contributed content. People can ask questions, answer them, and discuss a topic in this medium. The questions can range from "What should I wear to a prom?" to "How can I pseudo-inverse a matrix in Matlab?". Because of such big range of questions, most of the websites, such as Yahoo! Answers, maintain a categorization of them. However, sometimes the categorization is not enough to reduce the overwhelming number of threads a user can collaborate in. Recommender systems can be helpful in this situation. Although these resources can be used for item recommenders, most of the recommender systems that use forums and Q&A's data are internal: they aim to recommend interesting and related questions to the users or find participants for the questions. For example, Drod et al. [30] explore the problem of recommending questions in Yahoo! Answers to users. They use three levels of attributes from the forum data (textual, category, and social interaction attributes) to build a multi-channel recommender system. To add diversity to the recommended questions, Szpektor et al. [109] represent question and user profiles as a probability distribution tree, consisted of lexical vector of textual question contents, categorical distribution of the question in website's categories, and an LDA-based distribution of latent topics related to the question. Tang et al. [111] use the contents from user forums in a graphical model to predict the possible participants for some threads.

Micro-blogging:
Micro-blogging, such as tweeting or posting Facebook status, is real-time and fast in nature. As a result, it is a good resource to capture real-time events such as news. It is a resourceful medium to extract user opinions on almost everything, from a political campaign to a movie. In addition, the implicit and explicit structure of micro-blogging can be another resource of information for recommender systems. As for other user-contributed content, the recommendations that use micro-blogging can be internal or external: to recommend other user-contributed content [19] or to recommend people, items, places, etc., to users [3]. The recommendation approaches can be content-based [3], CF-based [27], or hybrid [19].

For example, Akther et al. [3] used a content-based and social recommender method that combine textual, social, and contextual information of users in twitter and Facebook to recommend items to users. Chen et al. [19] use content sources, topic interest models for users, and social voting to recommend interesting tweets to users. Sched et al. [100] use micro-blogs to find similarities between music artists. They use Twitter data to extract co-occurrence scores between music artists. One of their methods to evaluate these similarities is evaluating a content-based similar artist recommender system. Garcia and Amatriain [41] use features including content of the tweets to recommend followees to twitter users. Nasirifard et al. [86] introduce a follower recommender system that use follower's hashtags in Twitter to recommend them. Zangerle et al. [132] recommend hashtags to the Twitter user, based on the content that the user enters. De Francisci Morales et al. [27] use tweet contents with other features, such as demographic features, to build user profiles for rank-based recommendation of the news. Garcia et al. [42] use twitter real-time tweets of users to extract their opinions and perform movie recommendations. Guy et al. [50] use the activity stream of Lotus Connections and create personalized activity streams. Lotus Connections is a social media for organizations that includes a social networking system, a microblogging service, a bookmark application, a blogging application, forums, and a wiki system. TasteWeights [12] uses semantic data (from DBpedia [9]) with Twitter and Facebook data to recommend music to users.

Reviews and Comments:
User reviews and comments are common in websites such az Amazon and Yelp. The comments, as useful as they are, sometimes might get overwhelming. As a result, it is useful to have a recommender system that separates most useful reviews for a target user. In addition, textual reviews usually contain information about various aspects of an item. For example, in restaurant reviews, one can find information about the food, serving, noise, and other aspects of a restaurant. Aspect-based summarization can extract this information and recommend items to users based on the aspects that are important to them. The use of review information in recommender systems usually includes utilizing natural language processing methods such as sentiment analysis. For example, Leung et al. [74] apply natural language processing techniques to extract opinions and product features from user reviews on IMDB movies. They propose a frequency-based method to distinguish the sentiment orientation of the review and transfer them into ratings. Then, they use a CF approach to recommend movies to users. Ganu et al. [38] use a classification on user reviews (topics and sentiment) of the restaurants for recommendations. McAuley and Leskovec [84] combine latent rating dimensions of latent-factor recommender systems with latent review topics to predict items to users on Amazon. Garden and Dudek [43] extract semantic features of the movies from user comments and provide a hybrid recommender system of both content-based and collaborative filtering to users.

1.1.2 User-generated Lists

User-generated lists are groupings of items by users to make collections (e.g., music playlists, video playlists, categorization of scientific papers, etc.). These lists have the inherent categorization of items by user and the latent similarity information of the items in the list (in user's opinion). This similarity information can be useful for CF-based recommender systems and the categorization can be helpful in content-based and graph-based recommenders. Most of the music providers allow users to have playlists. Also, users can create lists in websites such as YouTube (for videos), CiteULike (for papers), etc. GroupMe! [2] is another application that allows users to arrange multimedia Web resources. Liu et al [81] also use hierarchical categories from the Open Dictionary Project (a hierarchical ontology scheme for site listings maintained by volunteer users) and past search behavior to represent a searcher?s profile.

There are several related work on using user article lists to recommend other articles, citations, co-authors, etc using Bibsonomy, CiteULike, Mendeley, or other bookmarking sites. For example Bogers et al. [11] use three different collaborative filtering algorithms on CiteULike data to recommend articles to users. Using music playlists as similarities in recommender systems is a little under explored as most of the recommender systems try to recommend playlists to users based on music tags [34] or the audio features of musical pieces [87]. As an example, Cohen and Web [21] use user playlists to perform a collaborative filtering recommendation of music. Arenas et al. [4] introduce a PLSA-based similarity fusion method to recommend music to users. Although they did not utilize the playlist similarity in their paper, they mention that this type of similarity can easily be embedded in their approach.

1.1.3 User Generated Multimedia

Youtube, Flickr, Facebook, Instagram, Sound Cloud, and photo blogs are examples of websites that include user-generated multimedia. Dealing with multimedia data is more difficult than other forms of user-contributed content as they need more sophisticated algorithms to generate features for them or classify them correctly. Most of the research on multimedia generated by users is on other features or user-contributed content related to them, such as tags or social relations. For example, Kazienko et al. [64] use the multidimensional social link structure of Flickr to recommend users to each other; or Zheng et al. [134] use Flickr tags to recommend groups to users.

While it might be a more difficult problem, there exist some approaches that use multimedia features along with other user-contributed data for recommendation. For example, Lindstaedt et al. [80] use tag propagation on visually similar images on Flickr to recommend tags. Lee et al. [73] propose visually weighted neighbor voting for learning the relevance of tags to images and recommend them. Abbasi et al. [1] propose to use tags as high level features, along with low level image features to create an image classifier on Flickr. Even though their task is not recommendation, their approach can be used for recommendations. Chen et al. [20] use visual and textual features of YouTube videos to find similar videos. Using two graph-based methods, they propagate the neighbor video tags in similar videos to recommend tags. Noulas et al. [87] use audio features of the musical piece to recommend playlists to users. Although they do not use user-contributed audio to learn the audio features, their method can be applied to user-generated audio as well.

As we can see, most of the recommender systems using user-contributed multimedia are internal: they recommend more user-contributed multimedia to users, or they recommend tags for them. There are some methods that perform external recommendation using photos, such as recommending landmarks based on user's Flickr photos [39]. But they mostly use image's geo-tags instead of photo features. In my research, I have not seen any papers on utilizing user-contributed video and audio externally. As example of external usage, I can think of recommending musical instruments or bands based on user generated music or recommending books based on user-contributed videos (specially videos of seminars and talks).

1.1.4 User's Geographical Data

Geotags in social networks have been very popular recently. Users provide their coordinates or location either by itself (e.g., Foursquare) or with other user-contributed content (e.g., geo-tagged photos). These information can be used in location based services and context-aware recommender systems in which the location of recommended item is important, such as places to visit, restaurants, music, etc.

For example, Freyne et al. [36] propose a mobile platform for tourist application that uses user location to recommend voice-based descriptions on a tourist spot. Shih et al. [104] design a system that extracts restaurants and food reviews from blogs to create a location-based recommendation system. Jiang et al. [62] use review data in addition to the community website and locations to build a local specialties recommender system. kurashima et al. [70] recommend travel routs based on a photographer's Flickr history by estimating the probability that the photographer visits a landmark. First, the photos are clustered based on their geotags. Then, user behavior is modeled with a combination of Markov model and topic model. Hsieh et al. [58] propose a method to incorporate four factors (popularity, visiting order, visiting time, and transit time) for recommending routs to users. They use user check-ins in the Gowalla dataset to train their probabilistic model. Levandoski et al. [75] design LARS, a hybrid recommender system that utilizes user partitioning and travel penalty to provide location-based recommendations. They experimented with Foursquare and MovieLens data. Yap et al. [127] provide a survey of user-generated content for location-based services.

1.2 Then using *cross-domain collaborative filtering* as the focus, discuss the *opportunities and challenges* of integrating user-generated contents into cross-domain recommendation.

Cross-domain collaborative recommendation focuses on providing recommendations in one domain based on usage information from other domains, e.g., recommending books based on user ratings on movies. However, some consider using external information such as social relationships to recommend items as another type of cross-domain recommendation [135, 99]. There are multiple ways to utilize other domain's information in cross-domain recommendation: from expanding the item space to both domains and applying collaborative filtering on the extended space [98, 123] to using a hybrid model of CF-based and content-based recommender system [110] to integrating the cross-domain information into the model of recommender system [82, 110]. This field is closely related to the domain adaptation problem in Natural Language Processing (NLP) with the difference that in NLP, the utilized data is textual or speech data and the goal is not exactly recommending itmes to users [33]. It is also related to the field of transfer learning [88] in machine learning, specially the methods that are applied to information retrieval problems [18]. Transfer learning has been applied to cross-domain recommender systems also [15, 77, 89].

1.2.1 The Opportunities

Access to more Cross-Domain Datasets One of the biggest challenges in cross-domain collaborative filtering is having access to cross-domain datasets. Although many commercial websites provide items from various domains to users, they do not easily share their datasets. Having this problem in addition to the fact that usage data is not always in the form of rating, makes user-contributed content a good source of information for cross-domain recommendations. For example, if we have user reviews on book domain but do not have user ratings or reviews in the music domain, we can use book reviews as a source of user interest and correlate the book interests with music ratings to recommend music.

Common Ground between NLP and Recommender Systems Among the various types of user-contributed content, text is one of the easiest types to embed in cross-domain collaborative filtering. Textual user-contributed data can be used in a common ground between NLP and recommender systems. For example, Mansour et al. [83] perform a theoretical analysis of the problem of domain adaptation and provide error bounds for weighted combining rule of domains. For experiments, they use domain adaptation on Amazon review data of four different domains: books, DVDs, electronics and, kitchen appliances. They label the reviews of one domain with the information of other domains. The results of this sentiment analysis can then be used as an auxiliary source of information in cross-domain recommendation. Also, the model can be integrated with a cross-domain recommender system. Joshi et al. [63] compare several domain adaptation methods to perform a similar task. They aim to find when multi-domain learning helps and what the good domains are for doing so using Amazon cross-domain reviews. Jakob and Gurevych [60] perform sentiment analysis in a cross-domain setting of reviews. They used IMDB, epinions.com, and another dataset including blog-postings for cameras to do the sentiment analysis although the objective was not cross-domain recommendation. Faridani [32] uses sentiment analysis on textual reviews. The author uses Canonical Correlation Analysis (CCA) to derive a mathematical model, used as multivariate regression, that uses both ratings and textual reviews. He uses Opinion Space (www.state.gov/opinionspace/) reviews to evaluate the model with Euclidean distance between each two users and their agreement rating. He does not provide any recommendation in his paper nor perform any domain adaptation; but the model can be extended to a cross-domain recommender system. There are other approaches to use sentiment analysis on review data for recommendation, but they are mostly in one domain such as the work cited in Section. 1.1.1 (e.g., Levi et al. [76] uses hotel reviews to provide hotel recommendations).

Relate Multiple Domains in Cross-Domain Recommender Systems: In addition to reviews, tags can be used to relate multiple domains in cross-domain recommendations. If two objects from different domains have the similar set of tags, they are more likely to be related/similar in general user opinion. For example, if a photo is tagged by "Napa" and "wine", it is more likely to be related to a hotel that is tagged by "Napa" and "winery". As an example of using tags in cross-domain recommenders, Wang et al. [120] introduce their model of tag transfer learning to use tags from multiple domains and provide recommendations. They use MovieLens dataset and divide it into two domains based on the density of tags in the data: auxiliary domain with denser tagging data and target domain with sparse tagging data. Then, they predict movie rating in the target domain with a topic-based similarity (obtained from tags) between users. Although their experiment does not include two real domains of items ans is on one domain separated into two, one can see the potential of their work in cross-domain recommenders. Shi et al. [103] use shared tags to link different domains using an extended matrix factorization framework. Here, the user-item matrices of different domains are transformed into domain-specific latent user and item features. Then, the latent features are linked across domains using tag-induced cross-domain similarities. Enrich et al. [31] also use auxiliary domain tags to recommend items in the target domain. They compare different models of transferring tags between the domains and experiment on MovieLens and LibraryThing datasets. They mostly focus on targeting the cold-start problem of cross-domain recommendations.

Combination of User-Generated Content Types: Different types of user-generated content can be used in combination with each other to recommend items or social relations. Authors in [64] use multiple relationships extracted from tags, opinions, groups, and lists of users to recommend social links to users.

Low Cost Real-Time Feedback and Other Opportunities: In addition to the opportunities mentioned above, user-contributed content introduces other opportunities for cross-domain and other types of recommendations. Users are voluntarily providing this data, so it needs low cognitive costs for the collectors. It provides real-time opinion of people on time-sensitive data and events, location or other context-related information of user, or the opportunity of having immediate feedback and communication with user. It lets us have a deeper understanding of user opinion and the reasons behind it. User-contributed content can help to understand individual needs of users. It also can provide a form of organization of objects with low cost and barrier. User-contributed content is scalable since we can access a vast amount of it and gathering this information has low computational cost. To be more specific to cross-domain collaborative filtering, as said before, the main advantage of them can be as a common medium between different domains.

1.2.2 The Challenges

Despite the potential of leveraging user-contributed content in cross-domain recommender systems, they are very under-explored. There are even relatively few publications on using textual user-contributed information in cross-domain recommender systems is very few. This can be attributed to the relatively new field of cross-domain recommender systems, lack of access to cross-domain user-contributed data, and the challenges in leveraging this data.

Reliability and Quality of the Data: There are several challenges in leveraging user-contributed data in the general form of recommender systems. First of all, user-contributed data is not always truthful. Users might provide incorrect information for many different reasons: because they do not feel like filling a form, they like gaming the system, they intentionally want to change other's (people or algorithms) opinions about the target item, or they like to hide the location they are at to protect their privacy. In addition, the quality of user-contributed content is not always high. The tags with spelling problems, reviews written in mix of different languages, spam blog posts and comments, and low-quality multimedia are the problems that occur in dealing with such data. Also, system design affects usefulness of user-generated data. For example, some of the tagging systems allow different users to tag one object multiple times with the same tag, while others allow one tag to be assigned to an object only once. While the second one can reduce the likelihood of gaming the system, the first method allows to define importance weights for the tags assigned the the object. Short forms of user-contributed content (such as tags and micro blogs) have their own problems: users cannot conduct many semantics using them. As a result they have little semantics while carrying many variations. There can be ambiguity in user-contributed content such as the problems of polysemy, synonymy, or different vocabulary usage for textual data. Specific to the tags, we should also deal differently with the tags that describe the object (e.g., "fiction" for movie) and the tags that explain user's opinion on that object (e.g., "cheesy" for movie).

There are some studies that target the problem of quality for user-contributed content. For example, Ye and Nov in [128] discuss that the quantity of contribution is inversely associated with average quality of contribution, but the inverse relationship between quantity and average quality of contributions is weakened by a longer tenure in the community. This study can provide some hints to how to use user-generated content: old users and the ones with less contribution are more trustable. There have been some efforts to filter high-quality user-contributed content. For example, Raghavan et al. [96] use review quality score to adjust the weight of review in their recommender system. They define the quality score based on text and metadata features of the review in Amazon (such as average rating of user).

Added Complexity to the System: Another issue with user-contributed content in recommender systems is the complexity they introduce to the system. CF usually suffers from scalability problems, specially in the memory-based versions that the whole input matrix or similarity matrix needs to be kept in memory. In case of having user-contributed content, we might need to keep more than one matrix (or in general more data) in the memory. In addition, user-contributed content adds to the computational expenses of the algorithm. It usually needs to be pre-processed before being used in the recommender system, or it is integrated to the model and adds to the dimensionality of space, number of parameters to be learned, number of features to be used, etc. As the complexity of the methods grow, we need more data to have acceptable statistical confidence in the results.

User Privacy: Privacy of users is another challenge that should be considered in gathering user-contributed content and using it in the recommender systems. Usually users can choose the people who can access their contents or have an expectation of who can see their content. A Facebook user can constraint the people who can see her photos or a twitter user thinks that he is only sharing with his followers when tweeting his opinions (not with some researchers storing the tweets or some algorithms). Also, the users might not expect the algorithm to extract information, other than the content they provided, from their generated content. For example, the user might not want the algorithm to extract and use the fact that he has a specific disease based on his comment on a medication. There are some related work in dealing with this problem. For example, Heitmann et al. propose an architecture for privacy-enabled user profile portability [55] and Heitmann proposes an open framework for privacy-aware cross-domain recommendations [54].

Heterogeneous Interpretations in Different Domains: Another issues is the challenge that is specific to cross-domain recommender systems: information might have different meanings and interpretations in different domains. When working with ratings data in cross-domain collaborative filtering, they almost always mean the same: in a 5-star likert scale, 5 stars is always good and one star is always bad. It is not the case for reviews on different domains: "cheesy" for a movie is a negative sentiment while it might be positive for a food; "unpredictable" is a positive book feedback while it is negative for electronic devices.

Heterogeneous User Interests in Different Domains: The last but not the least challenge related to cross-domain recommender systems is that user interests can be different in distinct domains. A user who likes "horror" movies might not exactly like "horror" books. As a result, we should note that using only the content information might be misleading in cross-domain recommendations. To target this issue, we can utilize the correlations in usage data, in addition to the content data, to achieve user interests in each of the domains.

To summarize, there are many opportunities and challenges to utilize user-contributed content in cross-domain recommender systems that should be explored more thoroughly in practice.

2 Educational Data Mining

Educational data mining is now increasingly used to **predict student performance** in solving problems and answering questions. A range of approaches have been proposed for that.

2.1 Choose at least *three* different approaches to performance prediction known to you. Present the approaches in a consistent way that makes it possible to compare them.

The goal of Predicting Student Performance (PSP) is to predict the student's capability to solve a problem or perform an educational task, mostly based on her performance in the past. It estimates measures of student's success, such as student's grade (a numerical value), or her degree of knowledge (a categorical value). PSP has been an active trail of research in educational data mining for a while and many machine learning and data mining approaches have been used to target this problem. Recently, this problem is getting more and more attention. In the KDD Cup 2010 Competition the focus was on prediction of students performance. PSP has been studied in the literature including bayesian networks [7, 90], logistic and linear regression [130, 17], decision trees [6], rule-based approaches [40], support vector machines [69, 112], neural networks [51, 44], and recently, latent space models such as models used in collaborative filtering [113].

Among the above approaches to PSP, I choose the following three to explain and compare:

- *Bayesian Knowledge Tracing (BKT)*
- *Performance Factor Analysis (PFA)*
- *Factorization Models*

2.1.1 Bayesian Knowledge Tracing (BKT)

Bayesian Knowledge Tracing (BKT) [23] is one of the pioneer methods in educational data mining. In this method, the student's knowledge on one Knowledge Component (KC) is modeled as a Markov Model with two states: either the student has learned the KC or not. As the student tries the same KC in a sequence, she can transition from the unlearned state to the learned state with learning probability $P(T)$ (Acquisition). The student's performance (usually in the form of giving the correct or incorrect answer to a question) is an observed variable. The student starts with an initial learning probability $P(L0)$. The probability of the student *guessing* the right answer for the question while she has not learned the KC is shown by $P(G)$ and the probability that the student *slips* and gives the wrong answer while she has already learned the KC is noted by $P(S)$. $P(L_n)$ shows the system's estimate of the current state and it is assumed that the student cannot go from a learned state to an unlearned state for a KC (e.g., cannot forget the skill). These parameters are estimated empirically from the data. After each attempt of the student, the knowledge state of the student is updated as follows:

$$P(L_n|action_n) = P(L_{n-1}|action_n) + (1 - P(L_{n-1}|action_n))P(T) \tag{1}$$

Equation 1 shows that the probability that a KC is learned in the n^{th} step is achieved by either the case that it has been already learned in the previous step, or it has not bean learned in the previous step, but is going to be learned in the transition. Using a Bayesian inference, after each step, the system updates its estimate of the student's knowledge as following:

$$p(L_{n-1}|Correct_n) = \frac{p(L_{n-1})(1 - P(S))}{p(L_{n-1})(1 - P(S)) + (1 - p(L_{n-1}))P(G)} \tag{2}$$

$$p(L_{n-1}|Inorrect_n) = \frac{p(L_{n-1})P(S)}{p(L_{n-1})P(S) + (1 - p(L_{n-1}))(1 - P(G))} \tag{3}$$

To learn the four parameters $P(L0)$, $P(G)$, $P(S)$, and $P(T)$, multiple methods such as curve fitting [23], Expectation Maximization (EM) [22], or brute force searching of space [93] are available.

For predicting the student's performance at the n^{th} step (the probability that a student j answers item i correctly at step n), we can use Equation 4.

$$P(C_{i,j}) = P(L_{k,j}) * (1 - P(S_k)) + (1 - P(L_{k,j})) * P(G_k) \tag{4}$$

In Equation 4, i represents the item (e.g., a question), k represents the knowledge component, and j represents the student. $P(S_k)$ and $P(G_k)$ are the slip and guess probabilities for KC k, and $P(L_{k,j})$ is the probability that the student j is in the learned state for the Knowledge Component k.

2.1.2 Performance Factor Analysis (PFA)

Performance Factor Analysis (PFA) [95] is an extension of Learning Factor Analysis [17] model. The goal in this model is to predict student's success based on her previous successes and failure on the same set of KCs and the difficulty of each item. The formulation of the model is as follows:

$$m(i, j \in KCs, k \in items, s, f) = \beta_k + \sum_{j \in KCs} (\gamma_j s_{i,j} + \rho_j f_{i,j}) \tag{5}$$

Here, m represents the accumulated learning for student i, β_k captures the easiness of each item, s shows the prior successes for the KCs for the student, f tracks the prior failures for the KCs for the student, and γ and ρ are the weights of these observation counts. The value m can be greater than one. In order to have a probability value for accumulated learning, we can use Equation 6.

$$P(m) = \frac{1}{1 + e^{-m}} \tag{6}$$

To find the parameters γ, β_k, and ρ, we can maximize the log-likelihood of the model on the provided dataset.

2.1.3 Factorization Models

Factorization models have been a strong tool in recommendation systems literature [67]. Recently, they have been used in the prediction student's performance [113]. In these models, the student and her success/failure or binary status of learning a KC are modeled in a matrix: each row shows a specific student i's record of the items/KCs and each column j shows how all of the students are doing with regards to one item/KC j. To capture the time effect in such a model, a 3-dimensional tensor (3-way array) can be used. In this case, each element of the tensor can represent the status of student i on item j in attempt k. To achieve a prediction of student's performance, we can use tensor factorization methods [114]. There are multiple approaches to do a tensor factorization on a tensor, which can be categorized in two main approaches: 1) as a sum of multiplication of one dimensional arrays (CANDECOMP/PARAFAC decomposition), 2) as a core tensor multiplied (or transformed) by a matrix along each mode (Tucker Decomposition) [65].

I further explain Bayesian Probabilistic Tensor Factorization (BPTF) [124] as a method for 3-way tensor decomposition. BPTF is a variation of CANDECOMP/PARAFAC (CP) decomposition that assumes the values change smoothly in the third dimension (time). So, the student's learning status in the k^{th} attempt is more likely to be similar to the student's $(k+1)^{th}$ attempt than her $(k+n)^{th}$ attempt on the same question ($n > 1$). This assumption can be translated into this interpretation that the student will learn and improve gradually during the attempts she is making. The model is as follows: if $R \in \mathbb{R}^{N \times M \times K}$ is the tensor representing the learning status of students $n \in 1..N$ on items $m \in 1..M$ in various attempts $k \in 1..K$, then we can approximate the value R_{ij}^k (learning status of student i on item j in attempt k) using three D-dimensional vectors U_i, V_j, and T_k using Equation 7. Here, U_i represents the i^{th} student's latent feature vector, V_j shows the j^{th} question's latent feature vector, and T_k is the latent feature vector for k^{th} attempt.

$$R_{ij}^k \approx < U_i, V_j, T_k > \equiv \sum_{d=1}^{D} U_{di} V_{dj} T_{dk} \tag{7}$$

To consider the randomness in the ratings, a Normal distribution is placed on R (Equation 8).

$$R_{ij}^k | U, V, T \sim N(< U_i, V_j, T_k >, \alpha^{-1}) \tag{8}$$

To avoiding over-fitting, we place a Gaussian prior distribution over U, V, and T. Also, we place a distribution over their hyper-parameters (mean and precision matrices) in order to have a non-parametric Bayesian estimation. To have a smooth learning transition between attempts, an additional constraint is added to approximation of T (Equation 9). This additional component imposes the predicted values of each time slice to be similar to the predicted values of its previous time slice with a small variation.

$$\begin{aligned} U_i &\sim N(\mu_U, \Gamma_U^{-1}), i = 1..N, \\ V_j &\sim N(\mu_V, \Gamma_V^{-1}), j = 1..M, \\ T_k &\sim N(T_{k-1}, \Gamma_T^{-1}), k = 2..K, \\ T_1 &\sim N(\mu_T, \Gamma_T^{-1}) \end{aligned} \tag{9}$$

We can learn the parameters by Markov Chain Monte Carlo. We can infer the predicted value for student's performance at various attempts on each item.

2.2 *Compare and contrast* the approaches by several aspects including *performance* and *applicability*, contrasting their *strong* and *weak* sides.

As I presented in Section. 2.1, there are several approaches to target the PSP problem. In the following, I will elaborate more on the characteristics of each of the approaches that I explained in Section. 2.1.

As seen in Section. 2.1.1, BKT explicitly models the probability of slip and guess. Using these latent factors in the model, captures the hidden causes of answering a questions correctly, while the student has not learned it or answering a question wrong, while the student has learned it before. It improves the accuracy and explanability of the model at the same time. Each of the four factors in BKT have a specific understandable meaning which makes sense and is understandable in explaining a sequence of user attempts. BKT uses the time factor in its model as a sequence of student's attempts and updates the model each time it gets new information about the student (observing the student trying an item). It helps the model to dynamically update itself and proceed to a more accurate model while maintaining its simplicity. On the other hand, personalization is a more difficult task in BKT. In the original form, it learns from all of the data for all of the students and the four factors that BKT learns are representative of the whole dataset. It assumes that all students have the same probability of knowing a particular skill at their first opportunity. Some attempts have been trying to personalize BKT by specializing the initial learning factor ($P(L0)$) for each student or having a specific $P(T)$ for each student [92], contextualization of the guess and slip parameters using a multi-staged machine-learning processes [24], or simply having an individualized factor for each student [23], but due to the complexity of the model and huge number of parameters to be learned, it is usually ignored. It needs many more data samples to be able to learn so many factors and also, it needs much more computational power. Even in the default setting of having only four factors for each of the KCs, BKT needs a good amount of computational resources to learn the factors. On the other hand, BKT can handle adding new students to the model easily. Since the parameters are not personalized, a new student can easily benefit from the BKT learned on other students' data and update the parameters as she attempts the items. Another problem of BKT is that it cannot consider multiple KCs at the same time. Each state represents the status of student in one KC and as a result, it is difficult to represent the items that have multiple KCs or skills in them. Having a binary state as learned and unlearned values is a simplification that sometimes might lead to less accuracy: knowledge can be a continuous value instead of a binary one.

As overviewed in Section. 2.1.2, PFA, unlike BKT, does not consider the probability of a student guessing or slipping while attempting an item. It only learns from the correct and incorrect practices and the difficulty of each item. It assumes that both correct and incorrect answers may help in learning the practiced skill (with different weights - γ and ρ). PFA does not model the temporal effect of the underlying process directly (as BKT does) but takes into account the prior success and failure of the student as a batch. Also, it can handle the case of multiple KC performances, where an item is consisted of multiple KCs at the same time. Since it uses a different γ and ρ weight for each KC, it can adjust the predicted performance based on how much a student knows in each of the KCs. In addition, lack of one KC can compensate for existence of another KC. Also, since it considers each student's success and failure separately in the model, it is easier for this model to predict a personalized learning performance for each student. However PFA has a disadvantage in personalization. It cannot distinguish between the value of success or failure of different KCs for different students: it supposes that having a success in a KC is as useful for student A as it is for student B. As for the new students, since there is no records of successes and failures of the student in hand, PFA can predict the performance based on the difficulty of the item. Finally, PFA is performed by calculating the gradiant to maximize the log-likelihood of the data; and as a result, has a fast optimization.

Factorization models, as explained in Section. 2.1.3, include the personalization factor inherently in their models. Both matrix factorization and tensor factorization models aim to predict a personalized value for the performance of each student on each item (skill, KC, etc). In the matrix factorization model, the predicted value is student's performance on each specific item in general. In the tensor factorization model, it is the student's performance on each item at each step. Because of

Model	Strength	Weakness
BKT	• Inherent modeling of time sequence • Explicit model of slip and guess factors • Ease of explanation to end user	• Computationally expensive • Difficult for personalization (individual students and KCs) • Cannot consider multiple KCs for each item/step • Binary modeling of learning state
PFA	• Learns from the prior correct and incorrect practice • Models the difficulty of an item explicitly • Can handle the case of multiple KC performances • Easier (than BKT) to personalize • Fast optimization to obtain the parameters	• Does not model the temporal effect of the underlying process directly • Does not consider slip and guess factors • Supposes that having a success in a KC is as useful for all student
Tensor Factorization	• Include the personalization factor inherently in their models • Easier (than PFA) to model time sequences • Can capture the correlation of KCs in the latent space • Can capture the difficulty of KCs in the latent space • Respectively fast	• Difficult to explain • Does not consider slip and guess factors • Do not consider KC compositions

Table 1: Strength/Weakness of the SPS approaches

Aspect	Feature	Worse	...	Better
Modeling and Performance	Computational Cost	BKT	Tensor Factorization	PFA
	Handling multiple KC cases	BKT	Tensor Factorization	PFA
	Explicit model of slip and guess factors	PFA	Tensor Factorization	BKT
	Modeling of time sequence	PFA	Tensor Factorization	BKT
	Capture the relationship between KCs	BKT/PFA		Tensor Factorization
	Models the difficulty of an item explicitly	BKT	Tensor Factorization	PFA
Applicability	Personalization of Performance	BKT	PFA	Tensor Factorization
	Ease of explanation to end user	Tensor Factorization	PFA	BKT

Table 2: Feature Comparison of the SPS approaches

dependency of these models on both students and items, it is difficult for some factorization models to cope with the new-student problem. In cases that the mapping between the original space and latent space is preserved, or in the generative model cases, we can expect the model to be able to predict the performance of a new student. In other cases, they cannot predict the performance of a new student. Since only the tensor factorization approaches can naturally include time in their model, I focus on these methods here. Note that although matrix factorization models cannot include time in their model inherently, we can embed the notion of time in them in the same way PFA does: by counting and normalizing the correct and incorrect attempts as a value for each cell or adding a time decay to this normalized value. If we look at the tensor factorization methods, we can see that not all of the methods take into account the notion of time as an ordered sequence. In most of the approaches, the third dimension is yet another categorical dimension (like the student id or item id) that can have different orderings without any specific change in the results. Consequently, we should be careful to pick only the methods of tensor factorization that are designed to consider the third dimension of the tensor as an ordinal value (like time slots). The BPTF method that was reviewed in Section. 2.1.3 does this by adding a constraint to the tensor factorization method. This constraint implies a smooth transition of values from each time slice to the next one. As a result, we can see that although most of the factorization methods do not model the underlying sequential nature of the data, they can do it better than, or as well as PFA.

Also the tensor factorization model of predicting student's knowledge is simple and understandable. Even though, it is difficult to explain the results of this model to the end user. The model does not consider the four factors of BKT. Instead, it approximates the complete student's performance tensor (including the future and missing attempts) using a summation of rank-one tensors (tensors that can be achieved by outer multiplication of three vectors). As a result, it maps all of the three dimensions of the tensor to one common latent space in which it can compare them directly. By summing the rank-one tensors that are achieved by multiplication of the vectors in this latent space, we can recover the complete tensor. As a result, while the new latent space and rank-one tensors do not add to the transparency of the approach, they can capture many hidden relationships in the data. One of the things that can be captured here is the relationship between KCs. Unlike PFA, tensor factorization cannot include multiple KCs into one item easily. However, these methods can capture the latent relationship between the KCs based on the correlation between them in the dataset. This correlation can be interpreted in multiple ways: whether the KCs are related to each other in the sense of the concept included in them, or they have the same level of difficulty for the students, etc. For this reason, these methods have been used in Q-learning problem in educational data mining [28]. Also, we can see that although these approaches do not model the difficulty explicitly, they are capable of capturing it in the latent space.

There have been some efforts to compare these three approaches in the literature. For example, Baker et al. compare BKT and PFA and their ensembles [25] and concluded that BKT is by far better than PFA; Pavlik et al. compared BKT and PFA [95] and concluded that PFA can do slightly better than BKT; Gong et al. compared multiple learnings of BKT and PFA [46, 47] and concluded that PFA and BKT don't have a significant difference in their predictive accuracy, but PFA's parameters were more plausible. They also mentioned that the model-fitting approach for BKT has a big effect on its parameters' plausibility. Lalle et. al compared the student models (and not the predicting performance) of BKT and PFA among other classifications [71] and measured them using their simulated impact on the success rate of tutorial decisions. They concluded that BKT does slightly better than PFA. Since using factorization models in predicting student's performance is new in the literature, to the best of my knowledge, there are no papers that compared these methods to BKT and PFA: Thai-Nghe et al. have proposed using factorized models [115, 114, 116, 113] and Toscher et al. used matrix factorization [117] for student performance prediction but in none of the works they have compared their models against BKT and PFA. As we can see, there are inconsistencies about which of BKT and PFA perform better in predicting students performance. The results depend on the dataset used and the approach used for obtaining parameters.

In summary, we can see the comparison and contrast of these three approaches in Tables 1 and 2. Table 1 explains the strong and weak sides of each method and Table 2 compares their applicability and performance. Since there are contradictions in the accuracy results of BKT and PFA, and since a non-published study of PAWS groups members, including me, suggests that the three methods perform comparably well, I did not include the accuracy as one of the measures.

2.3 Discuss prospects of *integrating* more than one different approach for performance prediction. Include both - known to your attempts to combine approaches and not yet explored ways of integration that you consider as possible and promising.

As seen in the sections 2.1 and 2.2, there are multiple approaches that can predict students performance, each of which has different strong and weak sides. One can think of integrating two or more of these approaches to benefit from their strength and compensate for their weaknesses. Since most of the approaches are based on classification, we can benefit from the existing methods of classification ensemble in the field of machine learning. Some of the methods to ensemble and integrate classifiers are mixture of experts, bagging, and boosting [8]. I call these methods "ensemble methods". In addition to these methods that integrate the results of multiple classification methods, we can think of forms of the integration that embed two or more ideas from each of the methods in one model and apply the new model to the data. I call this approach "model-based integration".

To summarize, I categorize the integrating methods as follows:

- *Ensemble Methods*
    - *Mixture of Experts*
    - *Committee Machines*
        - *Bagging*
        - *Boosting*
        - ...
- *Model-based Integration*

In the mixture of experts model, we try to fit separate classifiers for different parts of the data. So, we partition the data space according to the datapoint values and classify them separately. One of the basic models of mixture of experts is decision trees. In decision trees, we split the space recursively according to the input and assign the output label at the leaf of the tree. Another improved method is to use a combination of simple learners. Each classifier covers a specific region of the dataset and we put a smooth switch of classifiers between the regions so that they agree on the margins. In order to have the soft switch between the classifiers, we can use a gating function that decides which classifier to use for which of the datapoints. It can be implemented by a softmax model or a generative classifier model. As for its application to our problem, we can suppose that in some of the regions of our dataset (e.g., new users) the BKT model works better than PFA, and in other regions, PFA works better than BKT. Then we can use a mixture of experts in which each of BKT and PFA cover a different part of the input space. Even if we think that only BKT is sufficient to solve our problem, but different types of students need different parametrization of the model, we can use a mixture of BKTs, each learned on a different part of the dataset. In the PSP literature, the closest method to this idea has been published by Gong et al. [48] who clustered the students into k different clusters using k-means algorithm and then used a separate classifier on each of the clusters. Their approach is based on the assumption that students come from different distributions and so, need multiple classifiers. To the best of my knowledge, they do not have a soft switching model between the classifiers and also, they do not use different methods for each classifier.

Committee machine models try to use multiple base models (classifiers, regressors), each of which covers the complete input space. Each model is trained on a slightly different training set and the result of all of them are combined to produce the output. Among them, bagging models train multiple classifiers or regressors on the data and have the final prediction by averaging or majority voting over each of the models. Bagging helps in reducing the over-fitting of each of the models. But, if they suffer from a high bias, it cannot provide a better performance. On the other hand, boosting can solve the shortcoming of bagging: each of the learners focus on the regions that are not predicted well by the other classifiers. One of the pioneer methods of boosting is AdaBoost [35]. It classifies the data based on the weighted majority of the classifiers.

To learn the importance of each classifier, we need to evaluate on a validation set from the data. This ensemble can be easily related to the PSP problem. If each of the classifiers capture some of the information existing in the data, or if they are stronger in parts of the dataset, ensembling them using bagging and boosting should help to achieve a better result. We have tried a simple bagging model in an unpublished work by PAWS lab and we did not see any improvements as a result of ensemble. One explanation could be the bias that each of the models has. In this case, the boosting models should resolve this problem. Another reason can be the small amount of data that we had.

Most of the ensembling work in the current literature is related to boosting methods. For example, Baker et al. tried various methods for fusing various models of BKT with PFA and CFAR [25]. They used logistic, linear, and stepwise linear regression, with and without feature selection on the classification results. They found that a variation of BKT (BKT-PPS) gives the (non-significantly) best result in PSP when experiencing at the student level. In the action-level prediction (predicting each first attempt at each problem step in the data set), ensemble methods performed better than each of the single approaches. Later, Pardos et al. have explored several methods for ensembling PFA, BKT, and CFAR [91]. They used averaging, regression, AdaBoost, neural network, and random forest to fuse the three approaches. Most of the ensemble methods that they used worked better than single approaches both in action-level and student-level experiments. This is in contradiction with their previous work [25]. The reason behind this contradictory results appears to be the small size of the data in the previous work [25]. In KDD Cup 2010, Yu et al., who won the first prize, used ensemble of classification methods on various types of features [131]. They performed various ways of feature engineering before ensemble and ended up with two sets of condensed and sparse features. They used linear regression to find a weight vector to combine the predicted probabilities from each classifier. Kotsiantis et al. [68] provided online ensemble of classifiers that combine an incremental version of Naive Bayes, the 1-NN and the WINNOW algorithms using a voting methodology.

Another way of integrating multiple approaches, is the "model-based integration" by which I mean integrating various models into one combined model and have only "one" output for the whole model, instead of having one output per classifier and then combining them. A model-based integration approach that I think is worthy of exploring is Bayesian model averaging. In this model, we can explain the likelihood of the data by averaging over multiple models and learn the parameters of all of the models together. For example, if we have N different models m_i, $i \in 1 \ldots N$, we can write the likelihood of the data as follows (Equation 10):

$$P(D) = \sum_{i=1}^{N} P(D|M = m_i)P(M = m_i) \tag{10}$$

As a result, we can have the prediction as in Equation 11.

$$P(y|x) = \sum_{i=1}^{N} P(y|x, M = m_i)P(M = m_i) \tag{11}$$

There are a few studies in the literature that study the model-based integration. Xu and Mostow [125] use logistic regression in combination with BKT to improve the accuracy and the ability of the model to incorporate multiple KCs. Later, Xu and Mostow [126] used item response theory to improve knowledge tracing. Gonzalez and Mostow [49] provided a new approach that addresses both cognitive modeling and student modeling problems using graphical models.

To summarize, we can see that in some cases the ensembles and integrating approaches were very promising and in some, they did not do better than the single methods. There are only a few number of studies that have covered this area and in some of them, the datasets were very small in size. In general, the promising results of integration of classifiers in the machine learning area suggests that we need more investigation in this area of research in PSP.

3 Machine Learning

Suppose we wish to develop a diagnostic system for the Presbyterian emergency department (ED) at UPMC in Pittsburgh. Such a system might be used as a clinical aid to suggest plausible diagnoses in ED patients. We will assume a patient has **one primary disease** condition that leads him or her to visit the ED. The proposed system is intended to diagnose the primary disease condition. More specifically, we would like to develop a system that takes as **input patient ED findings** and generates as **output** a **probability distribution over n diseases** for that patient.

For data, we have 100,000 Presbyterian ED reports from the recent past. An ED report is a textual document that is generated by the primary ED physician who saw a patient. It typically contains a narrative description of the patient case, including a chief complaint, history, symptoms, signs, and current medications, as well as treatments in the ED and plans for the patient's subsequent care. While diagnostic impressions are generally stated in the report, let us assume here that they appear inconsistently, are not standardized terminologically, and are difficult to locate reliably in the report. Thus, for all practical purposes, we will assume that an ED report does not contain an explicit labeling of the primary diagnosis in a patient case. We will need to assess those labels from emergency medicine experts.

Assume we have a natural language processing (NLP) system that can reliably extract clinical concepts (variables) from an ED report and label each of them as currently being present or not. For example, if the report states ?the patient indicates she has a fever?, the variable fever would be assigned the value present by the NLP engine. Thus, each report is represented as a vector of concepts in which each concept is given one of the following values: (1) mentioned as present, (2) mentioned as absent, or (3) not mentioned (missing). Suppose the NLP engine identifies a total of 8,000 unique concepts over all 100,000 ED reports.

We would like to train the system as efficiently as possible to diagnose cases accurately. It is not feasible to have expert ED physicians review all 100,000 reports and label each case with one of the n diagnoses. Suppose it is practical, however, to ask them to review up to 10,000 patient cases and label them.

3.1 (a) Describe an approach to this problem that (1) uses latent Dirichlet allocation (LDA), a classification method of your choosing, and active learning to obtain labels from emergency medicine experts on up to 10,000 cases, and (2) constructs an ED diagnostic classifier based on those labeled cases. Provide enough detail and references that someone with graduate training in machine learning could take your description and construct the system you have in mind and write a paper about it without needing to locate additional material or references.

3.1.1 Preliminaries

Here, I briefly introduce LDA and Active Learning approaches.
Latent Dirichlet Allocation (LDA): Latent Dirichlet Allocation [10] is a generative Bayesian probabilistic model designed for discrete data such as textual data. It models the textual data as a set of documents consisted of a "bag of words". Each word in the document is generated by a hidden "topic" variable and each document has a distribution over the latent "topics". The plate model for the smoothed version of LDA is shown in Figure. 1.

In Figure. 1, w is one word in the document; θ is the distribution of k topics in the document; z

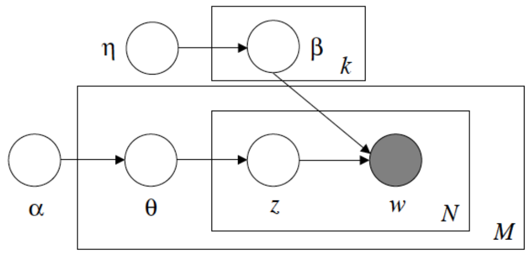

Figure 1: Smoothed LDA Graphical Model

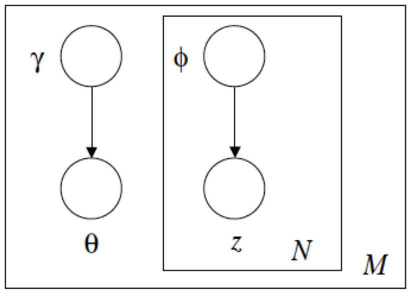

Figure 2: Variational Inference Representation for LDA

is the hidden topic of w; topics are assumed to be drawn from a Dirichlet distribution, $\beta_k \sim Dir(\eta)$; and α is the parameter of the Dirichlet prior on the per-document topic distributions. The true posterior distribution can not be computed directly and is usually approximated using Markov chain monte carlo (MCMC) or variational inference. In variational inference, the simplified model of Figure. 2 is used.

Active Learning: In active learning the learning algorithm can ask for the labels for some of the data samples interactively from the expert. Active learning is useful when we have a large number of unlabeled samples and we can afford to label a few of them. As a result, the samples that are given to user as a query should be selected precisely. The way we choose the desired samples is called querying strategy. Based on Settles's paper [101] the query strategies can be categorized as following:

1) Uncertainty sampling: label the unlabeled data on which the current model is most uncertain.

2) Query-By-Committee: pick the samples that have the most disagreement among several classifiers.

3) Expected Model Change: select the datapoints that would cause the greatest change to the current model if we knew their labels.

4) Expected Error Reduction: label data to minimize the expected error on the unlabeled data.

5) Variance Reduction: select the data that reduce the generalization error indirectly by minimizing output variance.

3.1.2 Mapping LDA to the Problem at Hand

As mentioned before, LDA originally was introduced to model textual documents. In our problem, we can consider each labeled ED report as a document and each of the present concept variables as a word seen in that document. So, a labeled ED report here is modeled using the symptoms of the patients as binary variables. If a symptom is present, the value of this binary variable is one, otherwise (unobserved or not-present) it is equal to zero. Since the multinomial distribution (on generating words for the document) handles the number of times a word occurs in a document, we cannot use it for not-present symptoms. This is a shortcoming of the model and I discuss it more sections 3.2 and 3.3. The topic distributions obtained for each ED report can be interpreted as either a distribution over possible diseases for that ED report or as a representation of the ED report that maps the report from the symptoms' space to a lower dimensional latent space. Some of the notation used in this question comes from LDA's notations: w is an observed symptom; z is a topic for the symptom; d is the ED report; and θ is the distribution of topics in an ED report.

3.1.3 Subsampling

I can think of multiple ways to construct a method, using LDA, for obtaining the required samples:

1. ***Hierarchical Sampling***: The first way is to use LDA to cluster the documents and ask for the labels that cover all of the obtained clusters from experts. In this method, we can run LDA, as an unsupervised method, on the whole set of $100,000$ data samples to obtain the topic distributions for each of the ED reports. Then, we can represent each ED report as a vector in the new lower dimensional latent space of topic distributions. The clustering can be based on various similarity measures between the ED reports in the new space; e.g., cosine similarity, L1 distance measure or Kullback Leibler divergence[1]. After performing the clustering, we can choose the active data samples (the data samples we are going to ask for a label from an expert) to be distributed in the clusters based on the size of each cluster. For example, if a cluster includes 30% of the datapoints, we choose 30% of the active data samples from this cluster. In this method, we are trying to reduce the expected output variance by sampling from all different clusters (hypothetically diseases) in the dataset. Although clustering using LDA will reveal the structure hidden in the dataset, it is a naive assumption to think that a clustering of the data has a one to one mapping to its actual labeling. Each cluster may contain labels from multiple different classes and the datapoints that are labeled with the same label might be scattered in different clusters. To target this problem and have an estimate of the error for uncertainty sampling we can use the work of Dasgupta and Hsu [26]. They used a hierarchical clustering of the data, guided by labels achieved by active learning so far, to cluster the data in a clustering that is representative enough of the underlying classification. In each iteration of hierarchical clustering, they calculate an empirical estimate of the error of a pruning (of the hierarchical clustering tree) by assuming that all datapoints of the cluster are assigned to the majority label of that cluster.

2. ***Informative Sampling***: The second approach is to find the irregular o novel data samples and choose them for labeling. My idea of irregularity comes from the work of Wang et al. [121]. Suppose that we want to know if a data sample (target datapoint) is irregular or not.

[1]Since we move to the simplex representation (distribution over topics), KL-divergence seems to be more suitable. But, the problem with KL-divergence is that it is no longer a metric.

We train an LDA (unsupervised) on the rest of the samples and find the parameters of the model based on those datapoints. Then, we calculate the probability that the trained LDA produces the target datapoint ($P(\mathbf{w}|\alpha, \beta)$). Here, $\mathbf{w}$ represents all of the observed symptoms in the target ED and α and β are trained based on the rest of the data. They are the priors that are learned in LDA (Look at Section 3.1.1). The lower this probability is, the more irregular the datapoint would be. As a result, the model is more uncertain about this datapoint. If we repeat the same process and get the $P(\mathbf{w}|\alpha, \beta)$ for all of the datapoints, we can sort them based on how irregular they are. Then, we can ask the experts to label the most irregular samples. The problem with this method is that we have to run LDA so many times on almost all of the datapoints. In our case, we have to run it $100,000$ times on datasets consisted of $99,999$ data samples. To reduce the computational cost, we can run LDA each time on a random sample of the data. But still the cost will be too much. Also, the generated set of irregular datapoints might be very biased and not representative of the whole dataset. Another issue with this approach is that it might find outliers instead of irregular samples. Since there are missing values (the unobserved and not-present symptoms), this might mislead the algorithm to mostly focus on outliers.

3. ***Residual Sampling***: Another approach is to use a variation of residual-LDA [119]. In their paper, Wahabzada and Kersting introduce an active method of scheduling of documents for batch and online LDA that reduces the required time for convergence and improves the model's perplexity and accuracy. This method schedules the documents that online-LDA receives in an online way: the schedule is not fixed and changes based on what it has been learned so far. In each iteration, residual-LDA selects the documents that have a large effect on the current residual to compute the next update. In simple words, in each iteration, they compute a measure of influence score $\rho(d)$ for all of the documents, sample S documents from $\rho(d)$, run one iteration of variational inference on the selected documents S, and update the influence score $\rho(d)$ for the next iteration based on the newly-learned parameters of variational inference for LDA. The influence score $\rho(d)$ is calculated based on the change of the variational parameter γ for each document (data sample) d in the iterations. If the parameter changes a lot, it means that the model is uncertain about that document and the document is picked earlier by LDA to train on. In the first iteration, the influence score is equal for all of the documents and it changes gradually as the algorithm iterates over the data. We can adopt this approach for selecting the ED reports that have the most influence on the learned LDA model as the active set of the samples. One of the problems with this method is that, as it updates the influence measure for all of the documents, it might pick the same document multiple times. To avoid this, we can discard the documents that have already been seen. Another problem of this approach for active learning is that it does not utilize the labels that have been given to the datapoints so far. It chooses the active samples in an unsupervised manner.

4. ***Hybrid Sampling***: The fourth approach is to use LDA to achieve a low-dimensional representation of the data in addition to a clustering of the data. Then, use multi-class SVM, initialized by the LDA clusters in the low-dimensional feature space, to find the active samples.

Based on the size of the dataset that we have and because of the big number of features in the dataset, I think the fourth approach (hybrid sampling) is a good fit to the problem. So, I choose to explain this method in detail. As said before, we can use LDA on the whole dataset of the ED reports to get a lower dimensional representation of the data. But, before that, to achieve the best number of topics k, we can separate a test and train set from the data, run LDA on the training set with varying number of topics and calculate the perplexity of the test data for the trained model. The lower the perplexity is, the better the model can predict the test data from the trained model and as a result the better the number of topics is. Then, we can run LDA on the whole set of the data with the picked number of topics. The distribution of each ED report document in each of the k topics of LDA can be obtained by $P(\mathbf{z}|d)$ (probability of the topics given the document). Using the low k dimensional representation of the data, we can cluster the samples the same way I explained in the first approach above (hierarchical sampling). As an example of the clustering method, we can use k-means clustering. Here, unlike hierarchical sampling, we do not need to worry about the samples from the clusters to cover the true classification labels because we are going to use multi-class SVM after this step. We can initialize the multi-class SVM with some initial samples from each cluster. Then, we use it to choose the active samples. Using LDA makes it easier for multi-class SVM to fit the classifier because the number of topics k is going to be much smaller than the number of original variables $(8,000)$. Also, mapping the data to lower-dimensional space might reduce the noise from the data (It should be tested by real data). Moreover initializing the multi-class SVM with LDA-based clusters can make the algorithm faster.

To use multi-class SVM for picking the best active samples in the lower dimensional representation of the data, we can follow the approach by Li et al. [78]. In their approach, Li et al. start by a random sample of the data and obtain the labels for them. Then, they initialize a multi-class SVM using these small set of data samples. After that, in each iteration, they choose the most informative samples for labeling and re-train the SVMs based on the new data labels. Suppose that u_s^{i*} is the set of selected samples. To choose the most informative samples in each iteration, we should calculate the decrease of the expected loss between two contiguous learning cycles (Equation 12).

$$u_s^{i*} = \arg\max_{u_s^i \in u^i} \left(\sum_{x \in I} L^i(x) - \sum_{x \in I} L^{i+1}(x, u_s^i)\right) \tag{12}$$

Here, $L^i(x)$ is the loss in iteration i for sample x; $L^{i+1}(x, u_s^i)$ is the loss in the $(i+1)^{th}$ iteration; and u^i is the set of unlabeled data in iteration i. Under some assumptions, the loss value for each datapoint is estimated as in Equation 13. The assumptions behind this estimation are that the training error is zero, the loss values of the documents used for training are zero, and all the expected losses of unselected unlabeled data have an equal influence between two contiguous learning cycles. Here, n is the number of classes predicted for x; $l_1, ..., l_n$ are the indexes of predicted classed for x; p is the total number of classes; m_{yj} is a component of a coding matrix M which is a $p \times p$ matrix with diagonal components 1 and others -1 (used as the true label of x); and $f_j(x)$ is the output of x on the j^{th} binary SVM classifier.

$$L(x) = \frac{1}{n} \sum_{y=l_1}^{l_n} \sum_{j=1}^{p} max[(1 - m_{yj} f_j(x)), 0] \tag{13}$$

$L(x)$ shows how much the classifier has improved during one iteration. So, the goal is to choose the datapoints that maximize $L(x)$. They modeled two selection strategies: Mean Max Loss (MML) and Max Loss (ML). In MML, the active datapoints (u_s^{i*}) are selected by maximizing $L(x)$ for the chosen datapoints:

$$u_s^{i*} = \arg\max_{u_s^i \in u^i} \sum_{x \in u_s^i} \frac{1}{n} \sum_{y=l_1}^{l_n} \sum_{j=1}^{p} max[(1 - m_{yj} f_j(x)), 0] \tag{14}$$

In the ML strategy, the loss value is calculated based on only the most certain predicted class label of the datapoint. So, the selected active set is defined based on Equation 15 in which y is the index of the most certainly predicted class of x.

$$u_s^{i*} = \arg\max_{u_s^i \in u^i} \sum_{x \in u_s^i} \sum_{j=1}^{p} max[(1 - m_{yj} f_j(x)), 0] \tag{15}$$

After selecting the active set in each iteration and getting the labels back, we can re-train the multi-class SVM for the next iteration.

For our problem, we can choose some number of random samples from each LDA-based cluster instead of the random sample initialization. Also, having the reduced number of features obtained by LDA makes every iteration of multi-class SVM faster and the initialization more accurate. Moreover, the number of classes in the first iteration can be estimated more accurately based on the LDA-based clustering.

3.1.4 Classification

After obtaining the $10,000$ samples, we can use classification algorithms to classify the rest of the dataset. We can classify the data using either multi-class SVM or semi-supervised LDA [121]. For using multi-class SVM, since the number of features are close to the number of labeled data instances, it is still better to use the low-dimensional representation of the $8,000$ features that are assigned by LDA.

3.1.5 Summary of the Proposed Method

I summarized the steps of the approach as following:

1. Run LDA on the whole set of data S;
2. Cluster the datapoints based on the topic distributions of the datapoints ($p(\mathbf{z}|d)$) using k-means clustering;
3. Pick s samples from each of the clusters to be labeled by the expert (represented in the original feature space);
4. Using the initial labels run multi-class SVM on the low-dimensional feature space of samples generated by LDA;
5. Train a semi-supervised LDA model using the labeled data achieved by step 4 and classify the rest of the data samples using that.

3.2 (b) What are the assumptions underlying your approach?

The first assumption that we have when performing the classification and active sampling is that the datapoints are IID (Independent and Identically Distributed). Since we use LDA to cluster the data, and then choose the multi-class SVM for picking the active samples, the final labeled data is not going to be independent nor identically distributed. The active datapoints are independent given the model that picked them, but not without conditioning on the model. We impose a sampling bias to the classifier we are going to use by using active sampling.

Another assumption in the proposed model is that the unobserved data and the not-present data are the same. Basically, when modeling with LDA, we are just considering the observed positive symptoms and ignoring the unobserved and observed negative symptoms. As a result, there is no difference between them from the model's point of view.

The third assumption is that the underlying structure of the data is revealed by the LDA-based clustering. Depending on the clustering method and the complexity of the data, this assumption might not be true.

The fourth assumption is that the cluster structure of the data reveals the classification partitioning of it. While it is a good heuristic, it does not have any proved theoretical reason.

A set of assumptions are related to the experts and patients. For example, we assume that the NLP method to extract disease symptoms work perfectly accurate. Also, we think that the symptoms are measured correctly in the ED reports and are not noisy. We assume the expert who labels the reports are perfect and don't make mistakes. Also, we assume that the patient has only one disease.

In addition to the above, the underlying assumptions of LDA, the chosen clustering algorithm, and multi-class SVM are part of the assumptions I have in the proposed approach. For example, LDA has the "bag of words" assumption: the order of the words in the document is unimportant to the model and the words are exchangeable. The Dirichlet priors in the LDA model are another set of assumptions that we have. We have the assumption that the relationships between ED reports and symptoms can be captured by LDA. We assume that the datapoints are random samples from the joint probability distribution we define in LDA. We have the assumption of discreteness for symptom values. We assume that the hyper parameters are independent of each other. In the multi-class SVM, we assume that the loss values of the documents used for training are zero (because of using proper kernels, the labeled documents can be linearly separable). Also, we assume that all of the expected losses of unselected unlabeled data have an equal influence between two contiguous learning cycles.

3.3 (c) Discuss the strengths and weaknesses of your approach.

One of the strong points of my proposed approach is that it addresses the "curse of dimensionality" in the data. By using LDA and moving the sample into a lower-dimensional space, the data samples that are given to the multi-class SVM have much lower number of features. In the cases where the data is not linearly separable, SVM moves the data into a higher dimensional space in order to be able to separate them linearly. Using the kernel trick, SVM can work in the original feature space to separate the classes non-linearly. For some of the kernels, even working in the original feature space needs an operation in order of the feature numbers or the dimensionality of the original space. For example, for polynomial kernel we have to calculate $\mathbf{x}^T\mathbf{x}'$ and for radial basis kernel we should calculate $\| \mathbf{x} - \mathbf{x}' \|^2$. Reducing $8,000$ features to the number of topics k will reduce the amount of time needed for these calculations.

Also, the strength that multi-class SVM brings into the approach is important. SVM is one of the best classifiers for linearly separable data. For the nonlinear data, although the training involves nonlinear optimization, the objective function is convex. As a result, it is straightforward to get the solution for optimization problem.

Another strength of the approach is having a starting point to initialize the active labeling. In other approaches, such as the original multi-class SVM for active sampling, the algorithms usually start by a random initialization of the active samples. The problem with random initialization is that the initial set might not include samples from all of the possible classes. For example, in a binary label case, such as spam detection, if 99% of the data are negative samples and 1% of the data are positive samples, we only have 1% of chance to obtain a negative sample with uniform random sampling. In addition, we miss the irregularities of the data by random sampling. Using clustering on the lower dimensional samples given by LDA, we get a better-than-random initialization for the multi-class SVM. Of course, the improvement of the approach versus the original multi-class SVM should be measured on real datasets and depends on the structure behind the data samples. But, we can say that the distribution of the initial samples obtained by LDA is closer to the underlying distribution of the data compared to a uniform distribution. As a result, initializing with these samples, reduces the sampling bias that other types of active learners may have. For example, if we use my second proposed method in Section 3.1.3 (separating irregular samples), the obtained samples are going to be biased towards the ones that cannot be modeled by the distribution over the rest of the dataset and the samples diverge from the original dataset.

Although the initial active samples for the multi-class SVM are less biased, the final label set that is achieved by multi-class SVM is still biased. This weakness can be seen in most of the active learning approaches. Here, the labeled set diverges from the whole distribution of the data to the samples that are most informative in placing the SVM hyperplanes in the dataset.

Another weak point of the approach is the complexity of it. It consists of three steps of running LDA, clustering the results, and running multi-class SVM. Although the first two steps reduce the complexity of the data and help multi-class SVM to perform faster than running multi-class SVM on the original features, each of the steps is computationally complex. Hence, obtaining the active samples might take long at the end.

The third weakness of my approach is ignoring some of the information in the dataset. As mentioned in Section 3.2, LDA ignores the not-present symptoms. As a result, it looses a useful source of information. One of the ways I can think of, to overcome this problem, is to extend the set of features. For each of the symptoms x, we can consider two binary variables: x_p and x_n. x_p equals to 1 if the symptom is present in the patient and equals to 0 if the symptom is not-present or is unobserved. x_n is equal to 1 if the symptom is not-present and equals to 0 either if the symptom is present or if it is unobserved. For example, if the patient has fever, $fever_p = 1$, and $fever_n = 0$. If the patient does not have a fever, $fever_p = 0$, and $fever_n = 1$. If we do not know about the fever of the patient, $fever_p = 0$, and $fever_n = 0$. There are two problems with this approach. First, the number of variables doubles and makes the classification more difficult. Specially here, the number of obtained features will be $16,000$ which is more than the number of labeled samples $(10,000)$. Also, if we want to use generative models, we have to add a handful of constraints to not to generate a sample with both variables equal to one.

As for the classification section (Section 3.1.4), using multi-class SVM to classify the unlabeled set of data has the weakness of not providing a probability distribution over the diseases. There have been some work on producing probability estimates based on SVM scores, but most of the obtained probabilities are not reliable [72]. On the other hand, if we use semi-supervised LDA, which is a generative model, we can get the probability distribution of the diseases for each of the unlabeled ED reports. Learning semi-supervised LDA is fast and easy because when we observe the class label (or topic z) for each of the samples (documents), the prior β and α will be independent given the observations. As a result, the parameters do not need to be estimated jointly any more and we can estimate them separately.

3.4 (d) Briefly suggest an alternative approach and indicate its strengths and weaknesses relative to your approach in part a.

An alternative approach can be active learning using multi-class SVM on the original data. Here, we do not apply LDA or cluster the data to get the active samples. So, we only have the steps 4 and 5 of the proposed hybrid sampling that I explained in Section 3.1.5 with the difference that The initial set for step 4 is a uniform random sample from the data. The expert labels this original set and gives it as the initial input to multi-class SVM. After finding an active set of samples and having the labels for them, we can use multi-class SVM or semi-supervised LDA to classify the rest of the data, as we did in Section 3.1.4.

Of the weakness of this approach is that the dimensionality of the data will be high and SVM classifier will have a hard time fitting on the data. Also, we have to initialize active sampling with a random set of the data that might not be representative of the whole dataset and can increase the number of samples we need to see in order to have a sample set with labels from all of the classes.

The strong side of this method is that we can naturally use both positive and negative-value samples in the SVM: each ED report can be represented by a vector of symptoms having the value $+1$ for the existing symptoms, -1 for the nonexistent symptoms and 0 for unobserved ones.

3.5 (e) Describe an experimental design for evaluating the classification performance of the ED diagnostic system that you developed in part a.

To evaluate the classification performance of the proposed approach, we can ask the experts to label a separate set of data and use it as the test set. Note that this set of labeled data should not come from the active learning algorithm that we have already used and should not have been seen by any of the steps of our classification. Then, we can run the trained classifier in Sections 3.1.3 and 3.1.4 and evaluate the accuracy of it by comparing the classification results with actual labels of the test data. We can calculate the confusion matrix for the results to check the classes in which the approach is weak or strong. The confusion matrix can also tell us about the class labels that are usually mistaken with each other by our approach. We can calculate the accuracy as follows:

$$\text{accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Number of test data samples}} \tag{16}$$

We can calculate the confusion matrix M as:

$$m_{i,j} = \text{the number of data samples classified as class } i \text{ that actually belong to class } j. \tag{17}$$

Another way to evaluate the classifier is to separate a test set from the data, not showing it to any of the algorithms in any steps. Then, calculate the perplexity of generating the test set using the model trained on the rest of the data. The lower the perplexity is, the better the model is in generating the test samples and the more generalizable the model would be.

Another method is to evaluate the approach by a disease retrieval task. We can use a labeled ED report as the query and ask the classifier to return the reports that has similar diagnosis as the result. Then, we can ask an expert to evaluate the returned reports and calculate the precision and recall for this task.

To evaluate the active sampling part of the approach, we can train multiple classifiers (including multi-class SVM and semi-supervised LDA) based on the generated active samples and test the accuracy of the classifiers on all of the classifiers.

Note that when we separate a uniform random set of samples for testing we might not see all the classes in the test set. Some of the diseases occur more often than others and their distribution is biased. Checking the model perplexity is a way to target this problem.

Also, note that the test set here is totally separate from what the active learner and classifier can see. Since we want to be fair to the algorithms, we cannot allow the active learner and the classifier to see these samples and learn from them. Even the active learner learns from the samples it sees to produce the active set of samples.

References

[1] R. Abbasi, M. Grzegorzek, and S. Staab. Using colors as tags in folksonomies to improve image classification. In *Proc. SAMTÕ08: Poster at Semantics And digital Media Technologies*, 2008.

[2] F. Abel, M. Frank, N. Henze, D. Krause, D. Plappert, and P. Siehndel. Groupme!-where semantic web meets web 2.0. In *The Semantic Web*, pages 871–878. Springer, 2007.

[3] A. Akther, K. M. Alam, H.-N. Kim, and A. El Saddik. Social network and user context assisted personalization for recommender systems. In *Innovations in Information Technology (IIT), 2012 International Conference on*, pages 95–100. IEEE, 2012.

[4] J. Arenas-García, A. Meng, K. B. Petersen, T. Lehn-Schioler, L. K. Hansen, and J. Larsen. Unveiling music structure via plsa similarity fusion. In *Machine Learning for Signal Processing, 2007 IEEE Workshop on*, pages 419–424. IEEE, 2007.

[5] P. Avesani, P. Massa, and R. Tiella. A trust-enhanced recommender system application: Moleskiing. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1589–1593. ACM, 2005.

[6] B. K. Baradwaj and S. Pal. Mining educational data to analyze students' performance. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 2(6):63–69, 2011.

[7] R. Bekele and W. Menzel. A bayesian approach to predict performance of a student (bapps): A case with ethiopian students. *algorithms*, 22(23):24, 2005.

[8] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[9] C. Bizer, S. Auer, G. Kobilarov, J. Lehmann, and R. Cyganiak. DbpediaÑquerying wikipedia like a database. In *Developers track presentation at the 16th international conference on World Wide Web, WWW*, pages 8–12, 2007.

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[11] T. Bogers and A. van den Bosch. Recommending scientific articles using citeulike. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 287–290, New York, NY, USA, 2008. ACM.

[12] S. Bostandjiev, J. O'Donovan, and T. Höllerer. Tasteweights: a visual interactive hybrid recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 35–42. ACM, 2012.

[13] R. Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[14] I. Cantador, M. Szomszor, H. Alani, M. Fernández, and P. Castells. Enriching ontological user profiles with tagging history for multi-domain recommendations. In *Proc. Collective Semnatics: Collective Inteligence and the Semantic Web*, page 5, 2008.

[15] B. Cao, N. N. Liu, and Q. Yang. Transfer learning for collective link prediction in multiple heterogenous domains. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 159–166, 2010.

[16] D. Caragea, V. Bahirwani, W. Aljandal, and W. H. Hsu. Ontology-based link prediction in the livejournal social network. In *SARA*, volume 9, pages 34–41, 2009.

[17] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *Intelligent Tutoring Systems*, pages 164–175. Springer, 2006.

[18] D. Chen, Y. Xiong, J. Yan, G.-R. Xue, G. Wang, and Z. Chen. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 13(3):236–253, 2010.

[19] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and tweet: experiments on recommending content from information streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1185–1194. ACM, 2010.

[20] Z. Chen, J. Cao, T. Xia, Y. Song, Y. Zhang, and J. Li. Web video retagging. *Multimedia Tools and Applications*, 55(1):53–82, 2011.

[21] W. W. Cohen and W. Fan. Web-collaborative filtering: Recommending music by crawling the web. *Computer Networks*, 33(1):685–698, 2000.

[22] A. Corbett, L. Kauffman, B. Maclaren, A. Wagner, and E. Jones. A cognitive tutor for genetics problem solving: Learning gains and student modeling. *Journal of Educational Computing Research*, 42(2):219–239, 2010.

[23] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[24] R. S. d Baker, A. T. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *Intelligent Tutoring Systems*, pages 406–415. Springer, 2008.

[25] R. S. d Baker, Z. A. Pardos, S. M. Gowda, B. B. Nooraei, and N. T. Heffernan. Ensembling predictions of student knowledge within intelligent tutoring systems. In *User Modeling, Adaption and Personalization*, pages 13–24. Springer, 2011.

[26] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215. ACM, 2008.

[27] G. De Francisci Morales, A. Gionis, and C. Lucchese. From chatter to headlines: harnessing the real-time web for personalized news recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 153–162. ACM, 2012.

[28] M. C. Desmarais. Conditions for effectively deriving a q-matrix from data with non-negative matrix factorization. In *4th International Conference on Educational Data Mining, EDM*, pages 41–50, 2011.

[29] J. Diederich and T. Iofciu. Finding communities of practice from user profiles based on folksonomies. In *EC-TEL Workshops*, volume 213. Citeseer, 2006.

[30] G. Dror, Y. Koren, Y. Maarek, and I. Szpektor. I want to answer; who has a question?: Yahoo! answers recommender system. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1109–1117. ACM, 2011.

[31] M. Enrich, M. Braunhofer, and F. Ricci. Cold-start management with cross-domain collaborative filtering and tags. In *E-Commerce and Web Technologies*, pages 101–112. Springer, 2013.

[32] S. Faridani. Using canonical correlation analysis for generalized sentiment analysis, product recommendation and search. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 355–358. ACM, 2011.

[33] J. R. Finkel and C. D. Manning. Hierarchical bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 602–610. Association for Computational Linguistics, 2009.

[34] C. S. Firan, W. Nejdl, and R. Paiu. The benefit of using tag-based profiles. In *Web Conference, 2007. LA-WEB 2007. Latin American*, pages 32–41. IEEE, 2007.

[35] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[36] J. Freyne, A. J. Brennan, B. Smyth, D. Byrne, A. F. Smeaton, and G. J. Jones. Automated murmurs: The social mobile tourist application. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 1021–1026. IEEE, 2009.

[37] K. A. Ganesan, N. Sundaresan, and H. Deo. Mining tag clouds and emoticons behind community feedback. In *Proceedings of the 17th international conference on World Wide Web*, pages 1181–1182. ACM, 2008.

[38] G. Ganu, N. Elhadad, and A. Marian. Beyond the stars: Improving rating predictions using review text content. In *WebDB*, 2009.

[39] Y. Gao, J. Tang, R. Hong, Q. Dai, T.-S. Chua, and R. Jain. W2go: A travel guidance system by automatic landmark ranking. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 123–132, New York, NY, USA, 2010. ACM.

[40] E. García, C. Romero, S. Ventura, and T. Calders. Drawbacks and solutions of applying association rule mining in learning management systems. In *Proceedings of the International Workshop on Applying Data Mining in e-Learning (ADML 2007), Crete, Greece*, pages 13–22, 2007.

[41] R. Garcia and X. Amatriain. Weighted content based methods for recommending connections in online social networks. In *Workshop on Recommender Systems and the Social Web*, pages 68–71, 2010.

[42] S. Garcia Esparza, M. P. O'Mahony, and B. Smyth. On the real-time web as a source of recommendation knowledge. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 305–308. ACM, 2010.

[43] M. Garden and G. Dudek. Mixed collaborative and content-based filtering with user-contributed semantic features. In *AAAI*, volume 6, pages 1307–1312, 2006.

[44] T. Gedeon and S. Turner. Explaining student grades predicted by a neural network. In *Neural Networks, 1993. IJCNN'93-Nagoya. Proceedings of 1993 International Joint Conference on*, volume 1, pages 609–612. IEEE, 1993.

[45] D. Godoy and A. Amandi. Hybrid content and tag-based profiles for recommendation in collaborative tagging systems. In *Latin American Web Conference, 2008. LA-WEB'08.*, pages 58–65. IEEE, 2008.

[46] Y. Gong, J. E. Beck, and N. T. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *Intelligent Tutoring Systems*, pages 35–44. Springer, 2010.

[47] Y. Gong, J. E. Beck, and N. T. Heffernan. How to construct more accurate student models: Comparing and optimizing knowledge tracing and performance factor analysis. *International Journal of Artificial Intelligence in Education*, 21(1):27–46, 2011.

[48] Y. Gong, J. E. Beck, and C. Ruiz. Modeling multiple distributions of student performances to improve predictive accuracy. In *User Modeling, Adaptation, and Personalization*, pages 102–113. Springer, 2012.

[49] J. P. González-Brenes and J. Mostow. Dynamic cognitive tracing: Towards unified discovery of student and cognitive models. In *EDM*, pages 49–56, 2012.

[50] I. Guy, I. Ronen, and A. Raviv. Personalized activity streams: sifting through the river of news. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 181–188. ACM, 2011.

[51] P. Halachev. Prediction of e-learning efficiency by neural networks. *Cybernetics and information technologies*, 12(2):98–108, 2012.

[52] C. Hayes. Using tags and clustering to identify topic-relevant blogs. In *Proc. 1st International Conference on Weblogs and Social Media (ICWSM 07)*. Citeseer, 2007.

[53] C. Hayes, P. Avesani, and S. Veeramachaneni. An analysis of bloggers and topics for a blog recommender system. In *Proc. WebMine 2006*, volume 4737 of *LNAI*, pages 1–20. Citeseer, 2007.

[54] B. Heitmann. An open framework for multi-source, cross-domain personalisation with semantic interest graphs. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 313–316, New York, NY, USA, 2012. ACM.

[55] B. Heitmann, J. G. Kim, A. Passant, C. Hayes, and H.-G. Kim. An architecture for privacy-enabled user profile portability on the web of data. In *Proceedings of the 1st International*

Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec '10, pages 16–23, New York, NY, USA, 2010. ACM.

[56] D. C. T. Hofmann. The missing link-a probabilistic model of document content and hypertext connectivity. In *Advances in neural information processing systems 13: proceedings of the 2000 conference*, volume 13, page 430. The MIT Press, 2001.

[57] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Folkrank: A ranking algorithm for folksonomies. In *LWA*, volume 1, pages 111–114. Citeseer, 2006.

[58] H.-P. Hsieh, C.-T. Li, and S.-D. Lin. Exploiting large-scale check-in data to recommend time-sensitive routes. In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, UrbComp '12, pages 55–62, New York, NY, USA, 2012. ACM.

[59] W. H. Hsu, A. L. King, M. S. Paradesi, T. Pydimarri, and T. Weninger. Collaborative and structural recommendation of friends using weblog-based social network analysis. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 55–60, 2006.

[60] N. Jakob and I. Gurevych. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1035–1045. Association for Computational Linguistics, 2010.

[61] J. Jiang, W. Pang, Y. Deng, K. He, and Z. Gu. A blog personality recommender system based on cloud computing infrastructure. In *Service Sciences (IJCSS), 2012 International Joint Conference on*, pages 1–5. IEEE, 2012.

[62] K. Jiang, L. Liu, R. Xiao, and N. Yu. Mining local specialties for travelers by leveraging structured and unstructured data. *Adv. MultiMedia*, 2012:15:15–15:15, Jan. 2012.

[63] M. Joshi, W. W. Cohen, M. Dredze, and C. P. Rosé. Multi-domain learning: when do domains matter? In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1302–1312. Association for Computational Linguistics, 2012.

[64] P. Kazienko, K. Musial, and T. Kajdanowicz. Multidimensional social network in the social recommender system. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(4):746–759, 2011.

[65] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[66] J. A. Konstan and J. Riedl. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1-2):101–123, 2012.

[67] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[68] S. Kotsiantis, K. Patriarcheas, and M. Xenos. A combinational incremental ensemble of classifiers as a technique for predicting studentsÕ performance in distance education. *Knowledge-Based Systems*, 23(6):529–535, 2010.

[69] A. Krištofič. Recommender system for adaptive hypermedia applications. In *IIT. SRC 2005: Student Research Conference*, page 229, 2005.

[70] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 579–588. ACM, 2010.

[71] S. LallÕ, J. Mostow, V. Luengo, and N. Guin. Comparing student models in different formalisms by predicting their impact on help success. In H. Lane, K. Yacef, J. Mostow, and P. Pavlik, editors, *Artificial Intelligence in Education*, volume 7926 of *Lecture Notes in Computer Science*, pages 161–170. Springer Berlin Heidelberg, 2013.

[72] A. Lambrou, H. Papadopoulos, I. Nouretdinov, and A. Gammerman. Reliable probability estimates based on support vector machines for large multiclass datasets. In *Artificial Intelligence Applications and Innovations*, pages 182–191. Springer, 2012.

[73] S. Lee, W. De Neve, and Y. M. Ro. Visually weighted neighbor voting for image tag relevance learning. *Multimedia Tools and Applications*, 67(3):1–24, 2013.

[74] C. W.-K. Leung, S. C.-F. Chan, F.-L. Chung, and G. Ngai. A probabilistic rating inference framework for mining user preferences from reviews. *World Wide Web*, 14(2):187–215, 2011.

[75] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. Lars: A location-aware recommender system. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 450–461. IEEE, 2012.

[76] A. Levi, O. Mokryn, C. Diot, and N. Taft. Finding a needle in a haystack of reviews: Cold start context-based hotel recommender system. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 115–122, New York, NY, USA, 2012. ACM.

[77] B. Li, Q. Yang, and X. Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 617–624. ACM, 2009.

[78] X. Li, L. Wang, and E. Sung. Multilabel svm active learning for image classification. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 4, pages 2207–2210. IEEE, 2004.

[79] H. Liang, Y. Xu, Y. Li, and R. Nayak. Collaborative filtering recommender systems using tag information. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 3, pages 59–62. IEEE, 2008.

[80] S. Lindstaedt, R. Mörzinger, R. Sorschag, V. Pammer, and G. Thallinger. Automatic image annotation using visual content and folksonomies. *Multimedia Tools and Applications*, 42(1):97–113, 2009.

[81] F. Liu, C. Yu, and W. Meng. Personalized web search by mapping user queries to categories. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 558–565. ACM, 2002.

[82] Y. Low, D. Agarwal, and A. J. Smola. Multiple domain user personalization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 123–131. ACM, 2011.

[83] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation with multiple sources. In *Advances in neural information processing systems*, pages 1041–1048, 2008.

[84] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 165–172, New York, NY, USA, 2013. ACM.

[85] A. K. Milicevic, A. Nanopoulos, and M. Ivanovic. Social tagging in recommender systems: a survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review*, 33(3):187–209, 2010.

[86] P. Nasirifard and C. Hayes. Tadvise: a twitter assistant based on twitter lists. In *Social Informatics*, pages 153–160. Springer, 2011.

[87] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil. An empirical study of geographic user activity patterns in foursquare. *ICWSM*, 11:70–573, 2011.

[88] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[89] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI*, volume 10, pages 230–235, 2010.

[90] Z. A. Pardos, J. E. Beck, C. Ruiz, and N. T. Heffernan. The composition effect: Conjunctive or compensatory? an analysis of multi-skill math questions in its. In *Proceedings of the International Conference on Educational DataMining*, pages 1147–156. Citeseer, 2008.

[91] Z. A. Pardos, S. M. Gowda, R. S. Baker, and N. T. Heffernan. The sum is greater than the parts: ensembling models of student knowledge in educational software. *ACM SIGKDD Explorations Newsletter*, 13(2):37–44, 2012.

[92] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *User Modeling, Adaptation, and Personalization*, pages 255–266. Springer, 2010.

[93] Z. A. Pardos and N. T. Heffernan. Navigating the parameter space of bayesian knowledge tracing models: Visualizations of the convergence of the expectation maximization algorithm. In *EDM*, pages 161–170, 2010.

[94] D. Parra-Santander and P. Brusilovsky. Improving collaborative filtering in social tagging systems for the recommendation of scientific articles. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 136–142. IEEE, 2010.

[95] P. I. Pavlik, H. Cen, and K. R. Koedinger. Performance factors analysis-a new alternative to knowledge tracing. In *AIEd*, pages 531–538, 2009.

[96] S. Raghavan, S. Gunasekar, and J. Ghosh. Review quality aware collaborative filtering. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 123–130, New York, NY, USA, 2012. ACM.

[97] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM, 2009.

[98] S. Sahebi and P. Brusilovsky. Cross-domain collaborative recommendation in a cold-start context: The impact of user profile size on the quality of recommendation. In *User Modeling, Adaptation, and Personalization*, pages 289–295. Springer, 2013.

[99] S. Sahebi and W. W. Cohen. Community-based recommendations: a solution to the cold start problem. In *Workshop on Recommender Systems and the Social Web, RSWEB*, 2011.

[100] M. Schedl, D. Hauger, and J. Urbano. Harvesting microblogs for contextual music similarity estimation: a co-occurrence-based framework. *Multimedia Systems*, pages 1–13, 2013.

[101] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[102] B. Shapira. *Recommender systems handbook*. Springer, 2011.

[103] Y. Shi, M. Larson, and A. Hanjalic. Tags as bridges between domains: Improving recommendation with tag-induced cross-domain collaborative filtering. In *User Modeling, Adaption and Personalization*, pages 305–316. Springer, 2011.

[104] C.-C. Shih, T.-C. Peng, and W.-S. Lai. Mining the blogosphere to generate local cuisine hotspots for mobile map service. In *Digital Information Management, 2009. ICDIM 2009. Fourth International Conference on*, pages 1–8. IEEE, 2009.

[105] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 515–522. ACM, 2008.

[106] J. Stoyanovich, S. Amer-yahia, C. Marlow, and C. Yu. Leveraging tagging to model user interests in del.icio.us. In *In AAAI SIP*, 2008.

[107] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 43–50. ACM, 2008.

[108] M. Szomszor, C. Cattuto, H. Alani, K. OÕHara, A. Baldassarri, V. Loreto, and V. D. Servedio. Folksonomies, the semantic web, and movie recommendation. In *Proceedings of the 4th European semantic web con- ference, bridging the gap between Semantic web and web 2.0*, pages 71–84, Innsbruck, 2007.

[109] I. Szpektor, Y. Maarek, and D. Pelleg. When relevance is not enough: promoting diversity and freshnessin personalized question recommendation. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1249–1260. International World Wide Web Conferences Steering Committee, 2013.

[110] J. Tang, S. Wu, J. Sun, and H. Su. Cross-domain collaboration recommendation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1285–1293. ACM, 2012.

[111] X. Tang, M. Zhang, and C. C. Yang. User interest and topic detection for personalized recommendation. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 442–446. IEEE Computer Society, 2012.

[112] N. Thai-Nghe, A. Busche, and L. Schmidt-Thieme. Improving academic performance prediction by dealing with class imbalance. In *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*, pages 878–883. IEEE, 2009.

[113] N. Thai-Nghe, L. Drumond, T. Horváth, A. Krohn-Grimberghe, A. Nanopoulos, and L. Schmidt-Thieme. *Educational Recommender Systems and Technologies: Practices and Challenges*, chapter Factorization techniques for predicting student performance. IGI Global, 2011.

[114] N. Thai-Nghe, L. Drumond, T. Horváth, A. Nanopoulos, and L. Schmidt-Thieme. Matrix and tensor factorization for predicting student performance. In *Proc. CSEDU'11*, pages 69–78, 2011.

[115] N. Thai-Nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme. Recommender system for predicting student performance. In *Proceedings of the 1st Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2010)*, volume 1 of *Procedia Computer Science*, pages 2811–2819. Elsevier, 2010.

[116] N. Thai-Nghe, T. Horvath, and L. Schmidt-Thieme. Context-aware factorization for personalized student's task recommendation. In *Proceedings of the International Workshop on Personalization Approaches in Learning Environments*, volume 732, pages 13–18, 2011.

[117] A. Toscher and M. Jahrer. Collaborative filtering applied to educational data mining. *Journal of Machine Learning Research*, 2010.

[118] K. H. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1995–1999. ACM, 2008.

[119] M. Wahabzada and K. Kersting. Larger residuals, less work: Active document scheduling for latent dirichlet allocation. In *Machine Learning and Knowledge Discovery in Databases*, pages 475–490. Springer, 2011.

[120] W. Wang, Z. Chen, J. Liu, Q. Qi, and Z. Zhao. User-based collaborative filtering on cross domain by tag transfer learning. In *Proceedings of the 1st International Workshop on Cross Domain Knowledge Discovery in Web and Social Network Mining*, pages 10–17. ACM, 2012.

[121] Y. Wang, P. Sabzmeydani, and G. Mori. Semi-latent dirichlet allocation: A hierarchical model for human action recognition. In *Human Motion–Understanding, Modeling, Capture and Animation*, pages 240–254. Springer, 2007.

[122] R. Wetzker, W. Umbrath, and A. Said. A hybrid approach to item recommendation in folksonomies. In *Proceedings of the WSDM'09 Workshop on Exploiting Semantic Annotations in Information Retrieval*, pages 25–29. ACM, 2009.

[123] P. Winoto and T. Tang. If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? a study of cross-domain recommendations. *New Generation Computing*, 26(3):209–225, 2008.

[124] L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222, 2010.

[125] Y. Xu and J. Mostow. Using logistic regression to trace multiple sub-skills in a dynamic bayes net. In *EDM*, pages 241–246, 2011.

[126] Y. Xu and J. Mostow. Using item response theory to refine knowledge tracing. In *EDM*, 2013.

[127] L. F. Yap, M. Bessho, N. Koshizuka, and K. Sakamura. User-generated content for location-based services: A review. In *Virtual Communities, Social Networks and Collaboration*, pages 163–179. Springer, 2012.

[128] C. Ye and O. Nov. Exploring user contributed information in social computing systems: quantity versus quality. *Online Information Review*, 37(5):752–770, 2013.

[129] A. Yeung, C. Man, N. Gibbins, and N. Shadbolt. A study of user profile generation from folksonomies. In *social web 2008 workshop at WWW2008*, 2008.

[130] C. Yu, A. Jannasch-Pennell, S. Digangi, and B. Wasson. Using online interactive statistics for evaluating web-based instruction. *Journal of Educational Media International*, 35:157–161, 1999.

[131] H.-F. Yu, H.-Y. Lo, H.-P. Hsieh, J.-K. Lou, T. G. McKenzie, J.-W. Chou, P.-H. Chung, C.-H. Ho, C.-F. Chang, Y.-H. Wei, et al. Feature engineering and classifier ensemble for kdd cup 2010. In *Proceedings of the KDD Cup 2010 Workshop*, pages 1–16, 2010.

[132] E. Zangerle, W. Gassler, and G. Specht. Recommending#-tags in twitter. In *Proceedings of the Workshop on Semantic Adaptive Social Web (SASWeb 2011). CEUR Workshop Proceedings*, volume 730, pages 67–78, 2011.

[133] X. Zhao, J. Yuan, M. Wang, G. Li, R. Hong, Z. Li, and T.-S. Chua. Video recommendation over multiple information sources. *Multimedia Systems*, 19(1):3–15, 2013.

[134] N. Zheng, Q. Li, S. Liao, and L. Zhang. Which photo groups should i choose? a comparative study of recommendation algorithms in flickr. *Journal of Information Science*, 36(6):733–750, 2010.

[135] E. Zhong, W. Fan, J. Wang, L. Xiao, and Y. Li. Comsoc: adaptive transfer of user behaviors over composite social network. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 696–704. ACM, 2012.