Incorportaing Content and Context in Recommender Systems

by

Torin Stepan

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Recommender systems are a growing area of research that find practical applications in a variety of domains. Integrated library systems and location-based social networks can apply recommendation algorithms to assist their users in finding an item or location that suits their needs. With an ever-increasing variety of options to choose from, deciding on which book to read or movie to watch can become overwhelming. Recommender systems aid their users in the decision making process by providing a list of items likely to be relevant to the user's needs and interests. A persistent issue faced by recommender systems is a lack of data concerning the preferences of its users, known as the "cold-start" problem, which leads to poor recommendation quality, particularly for new users and items. To improve recommendation quality in the face of incomplete data, we propose several novel approaches for incorporating all available data into collaborative filtering algorithms.

Preface

Chapter 3 of this thesis has been submitted to IEEE Transactions on Fuzzy Systems as Stepan, T., Dick, S. & Miller, J. (2014). A Fuzzy Recommender System for Public Library Catalogs. Chapter 4 of this thesis has been submitted to ACM Transactions on Information Systems as Stepan, T., Dick, S. & Miller, J. (2014). Incorporating Spatial, Temporal, and Social Context in Recommendations for Location-Based Social Networks. Parts of chapters 1 and 2 of this thesis are also present in these submissions. For both of these papers, I have conducted the entirety of the research and experimentation, with editorial assistance and guidance from my supervisors, Scott Dick and James Miller.

Acknowledgements

I wish to thank my supervisors, Scott Dick and James Miller, for their continued support and guidance throughout the entirety of my time spent at the University of Alberta in both undergraduate and graduate studies. I would also like to thank my professors, particularly Marek Reformat, Lukasz Kurgan, and Witold Pedrycz for their engaging lectures that helped form some of the ideas presented in this thesis.

I would not have been able to succeed without the support of my friends, family, and colleagues. Jeremy Kerr-Wilson encouraged me to pursue graduate studies in the first place, and he, along with many others, helped keep me sane throughout the long process of research, experimentation, writing, rewriting, and rewriting again. To my family and friends -- Nevin Stepan, Quinn and Elise Barber, Ryan Hemphill, Neale Wiley, Sepehr Parhami, Joseph Pallechio, Tim McKinney, Morris Lo, Wayne Poon, Melissa Rodenburg, Victoria Merritt, Rick VanManen, Ken and Shelley Stepan -- thank you all.

Table of Contents

List of Tables

List of Abbreviations

CF	-	Collaborative Filtering
FBCA	-	Friendship-based Bookmark-Coloring Algorithm
ILS	-	Integrated Library Systems
KNN	-	K-Nearest Neighbors
LBSN	-	Location-Based Social Network
LFBCA	-	Location-Friendship Bookmark-Coloring Algorithm
LRT	-	Location Recommendation framework with Temporal effects
MAE	-	Mean Absolute Error
PPR	-	Personalized Page Rank
RMSE	-	Root Mean Squared Error
TF-IDF	-	Term Frequency/Inverse Document Frequency

1 Introduction

In 2012, worldwide e-commerce sales accounted for over 1 trillion dollars of consumer spending [1]. E-commerce is now a vital part of the world economy, but this success also creates new problems. One of the most important problems is discovery; with literally a world of options to choose from, it is extremely difficult for e-commerce consumers to find the product that best fits their needs [2]. Recommender systems attempt to solve this problem by learning to predict which products a consumer is likely to find relevant, using both the consumer's own consumption patterns and those of "similar" consumers. These highly-relevant products are then presented to the consumer as suggestions. This is a high-value niche within the intelligent systems field; recommenders are known to significantly increase e-commerce sales [3], with Netflix and Amazon.com both claiming that their recommenders generate large fractions of their revenues streams [4]. Recommenders are generally designed for a specific industry; clearly, a recommender designed to suggest pay-per-view movies will be of little help in selecting a restaurant. In this thesis, we study how recommendation algorithms can be applied to the domain of integrated library systems (ILS) and location-based social networks (LBSNs).

Location-based social networks (LSBNs) are a growing domain with ever-increasing amounts of smartphone users utilizing location-based services [5] [6]. Geo-social services, such as Foursquare, allow a user to check-in through a mobile device at a location of interest such as a restaurant, mall, or movie theatre. A user performing a check-in notifies their online friends of their location and provides an opportunity to interact. Foursquare is growing substantially: in April, 2012, it had around 20 million users and 2 billion user submitted check-ins [7]; as of May, 2014, its community has grown to over 50 million users and 6 billion check-ins [8]. With

thousands of potential points of interest in the vicinity of each user, the process of choosing a new location to visit can be overwhelming [9] [10] [2]. Recommendation algorithms strive to address this issue by filtering through the staggering variety of options available and returning those most likely to be of interest.

Public libraries can play a particularly important role in their communities, and especially in rural and northern Canadian communities. As a very general statement, these communities tend to be remote and sparsely populated (in 2010, the population of the Canadian North was 2.1 million people, in a land area twice the size of India [11]). Demographically, they tended to be younger than major Canadian cities, with higher proportions of Aboriginal residents. Many of these communities have very limited job opportunities, and educational and health outcomes are noticeably worse than in urban centres. Partly this is due to a lack of resources, and partly due to legacy effects from government policies (e.g. the experience of the residential school system has left many Aboriginal people with a deep-seated distrust of formal education [11]). While online learning has the potential to address the lack of school resources, establishing and maintaining broadband Internet connections for remote Northern communities at an affordable cost is also not trivial. From this, we see that the *social capital* (broadly defined as the benefits conferred on individuals and the group by participating in social relationships [12]) of these rural communities is weaker than in urban centres. The public library is one of the key social capital resources in these communities; it provides resources for learning and discovery, and a place to connect with others having similar interests [11]. Thus, increasing residents' engagement with the library is one important route to increasing the community's social capital, and recommender systems have been shown to have a positive effect on engagement [13]. Thus, the development of

recommender systems for rural libraries is a potentially significant step in furthering the social development of these communities.

Traditional approaches to recommendation include collaborative filtering, content-based recommendation, and hybrid methods. More recently, context-aware recommender systems [14] have gained popularity. User-user collaborative filtering [15], [16] recommends items to a user based on the items preferred by similar users, where similarity between users is defined by the degree of overlap between the rating or purchasing histories of two users. Item-item collaborative filtering [17] behaves in a similar way as the user-based variant, recommending items similar to those that the user has preferred in the past. Content-based approaches [18] [19] leverage data about items beyond the ratings provided by users, including attributes such as genre information, tags, or authors. Context-aware recommendation is a generalization of content-based recommendation, and considers a wider array of circumstantial data. Context can be defined as the state of the user, consisting of data such as the time, weather, or location [20]. The common goal of all recommender systems is that they must endeavor to generate accurate recommendations from incomplete information on the preferences of users and attributes of items. This is a challenging task, particularly for users and items with few ratings. The inability of recommender systems to produce accurate suggestions due to insufficient data is known as the cold-start problem [21]. In large-scale operations (e.g. Amazon.com), the volume of customers and the breadth of available holdings makes it likely that, as a new customer engages with the system, their histories begin to match other customers reasonably well. In other words, they join the gestalt that emerges from the combined reading interests of a huge number of customers. Likewise, as a new item is rated by more users, it too becomes part of this gestalt. In the context

of a rural library, however, it is reasonable to expect that the cold-start problem might never be resolved; these library will have very limited holdings, and very limited clienteles, and so a patron's reading history might never be adequately similar to other patrons'. In other words, a gestalt might emerge only for a small subset of the library's most active patrons, those who are already strongly engaged with the library. If this is the case, then the recommender is unlikely to generate helpful recommendations for the other users, thus suppressing the anticipated increase in engagement with the library. Thus, our problem is to create a public-library recommendation algorithm that can provide accurate results even in the face of this persistent cold-start problem.

For integrated library systems, we propose to hybridize an item-item collaborative filter with the addition of content data, and a fuzzy *taste vector* [22] generated from that content data. Thus, rather than attempting to discover (for example) genre or author preferences implicit in the itemsets borrowed by various users, these preferences can be directly inferred from the meta-data (e.g. MARC records) in the card catalog. We use the fuzzy taste vector technique as it allows us to handle non-binary ratings given by users, translating them into a representation of the degree to which a user prefers each genre. We evaluate our new algorithm on two well-known recommendation benchmarks: Book-Crossing [23] and MovieLens [24]. The results show that our algorithm is at least as accurate as any other fuzzy recommender on these datasets, and offers superior prediction coverage.

To apply to the domain of LBSNs, we propose a framework which leverages contextual data in order to improve the quality of recommendations generated by a collaborative filtering

algorithm. We incorporate temporal, social, and spatial context piece-by-piece into a collaborative filter and examine the impact of including each layer. Our proposed method is evaluated experimentally on three datasets collected by Gao et al. [18] (Foursquare) [25], as well as two datasets from Cho et al. [26] (Brightkite and Gowalla). The results of our experiments show that the predictive accuracy of our approach is superior to that of the previous research on these datasets.

The remainder of this thesis is organized as follows. In Chapter 2 we review traditional collaborative filtering techniques, and discuss how content-based and context-aware approaches can extend basic collaborative filtering. Chapter 3 describes related work in the field of fuzzy recommendation algorithms, particularly for ILS, and details our approach to recommendation for rural libraries. We present our work on context-aware recommendation for LBSNs in chapter 4. In Chapter 6, we conclude with a summary and discussion of future work.

2 Collaborative Filtering Recommendation Systems

2.1 Collaborative Filtering

2.1.1 The recommendation problem

Recommender systems strive to predict what a user would find interesting. Given a set of users, U, and a set of items, I, a recommender system attempts to find the subset of items that are the *most relevant* to each user (U_j) [27]. An item can be anything that the user can interact with, for example, a book, movie, store, or event. A user's *item history* (I_{U_j}) -- the items which they have previously interacted with -- can give some insight into the items that they may find interesting in the future. A user's item history may contain detailed rating information ($I_{U_j} \rightarrow R_{U_j} = r(U_j, I_{U_j})$) (for example, a value on a scale of 1-5), or it may be as simple as an indication that a user has purchased an item without any additional feedback (0 or 1). The combined item history of all users is referred to as the *user-item matrix* $\bigcup_j U_j \times I_{U_j}$. Collaborative filtering algorithms utilize the data present in the user-item matrix to predict how a particular user would rate an item they have not yet interacted with. In our discussion of data used in recommender systems, we use the following terms [16]:

- Explicit rating: a rating given by a user on a scale (typically 1-5 or 1-10) indicating their level of preference for a particular item.
- Implicit rating: an indication that a user has purchased/read/used an item, but has not provided an explicit rating for it.
- Sparsity: the fraction of possible ratings that have not yet been assigned. This is defined by:

$$sparsity = \frac{|U||I| - |R|}{|U||I|} \tag{2.1}$$

where U is the set of all users, I is the set of all items, and R is the set of all ratings that have been provided to date.

2.1.2 Similarity Measures

User-based collaborative filtering operates under the premise that the future behaviour of a user can be predicted from the past behaviour of similar users [15], [16], [28]. The similarity between users is based on the degree to which their past ratings or interactions agree. Various methods for quantifying the similarity between users or items in a recommendation system exist; a popular similarity measure is Pearson's correlation coefficient [16] [29], which is calculated as follows:

$$sim(U_j, U_k) = \frac{\sum_{i \in C} \left(r(U_j, i) - \bar{r}(U_j, I_{U_j})\right)\left(r(U_k, i) - \bar{r}(U_k, I_{U_k})\right)}{\sqrt{\sum_{i \in C} \left(r(U_j, i) - \bar{r}(U_j, I_{U_j})\right)^2 \left(r(U_k, i) - \bar{r}(U_k, I_{U_k})\right)^2}} \tag{2.2}$$

$C = I_{Uj} \cap I_{Uk}$ represents the set of items that have been rated by both users, $r(U_j, i)$ represents the rating given by user U_j to item $i \in I$, and $\bar{r}(U_j, I_{U_j})$ represents the average rating user U_j has given to all items in their item history.

A possible downside of the Pearson correlation coefficient is that it does not place any weight of the size of the set of co-rated items. For example, consider a case with three users: *A, B,* and *C*, each of whom has rated *n* items and has the same mean rating. Let user *A* and user *B* generally (but not exactly) be in agreement on *n-1* co-rated items. Then, let user *C* co-rate only 1 item with

user *A*, but with the exact same rating. The Pearson's correlation coefficient would determine that user *A* is more similar to user *C* than they are to user *B*, despite the fact that there is more data to support a link between users *A* and *B*. To give credit to a larger set of co-ratings, we scale Pearson's correlation coefficient by the Jaccard similarity index [30]:

$$sim_{Jaccard}(U_j, U_k) = sim(U_j, U_k) \frac{\left| I_{U_j} \cap I_{U_k} \right|}{\left| I_{U_j} \cup I_{U_k} \right|} \tag{2.3}$$

where $sim(U_j, U_k)$ is defined in Eq. (2.2), the numerator is the cardinality of the set containing all items co-rated by users U_j and U_k, and the denominator is the cardinality of the set of items that either one of U_j or U_k have rated.

Cosine similarity is another popular similarity measure [31]. Treating the item histories of two users as vectors, I_{U_j} and I_{U_k}, we find their similarity as follows, using the dot product and magnitude:

$$sim(U_j, U_k) = \frac{I_{U_j} \cdot I_{U_k}}{\left| I_{U_j} \right| \left| I_{U_k} \right|} \tag{2.4}$$

This similarity measure has the advantage of naturally accounting for the degree of overlap between two users. If two users share a small number of co-ratings in relation to the magnitude of their item histories, their cosine similarity will be low. Thus, scaling this similarity metric by the Jaccard similarity index is not needed to avoid dissimilar users whose ratings happen to agree on a small set of items being considered similar.

Other similarity metrics such as the Hamming distance [32] or mean squared difference [27] have been applied to user-based collaborative filtering. Essentially, the goal of a similarity metric in a collaborative filtering environment is to find sets of users that can best predict the future behaviour of each other.

2.1.3 User-based vs. item-based collaborative filtering

In the previous section, we discussed similarity metrics for user-based collaborative filtering. Item-item collaborative filtering [17], [18] functions much the same as the user-based variant, but by calculating similarities between items rather than users. For an item-based collaborative filter, an equivalent similarity measure to Eq. (2.2) can be defined as the adjusted cosine similarity:

$$sim\left(I_j, I_k\right) = \frac{\sum_{u \in C} \left(r\left(u, I_j\right) - \bar{r}\left(I_j, U_{I_j}\right)\right)\left(r\left(u, I_k\right) - \bar{r}\left(I_k, U_{I_k}\right)\right)}{\sqrt{\sum_{u \in C} \left(r\left(u, I_j\right) - \bar{r}\left(I_j, U_{I_j}\right)\right)^2 \left(r\left(u, I_k\right) - \bar{r}\left(I_k, U_{I_k}\right)\right)^2}} \tag{2.5}$$

$C=U_{Ij} \cap U_{Ik}$ represents the set of users that have rated by both items, $r(u,I_j)$ represents the rating given by user $u \in U$ to item I_j, and $\bar{r}\left(I_j, U_{I_j}\right)$ represents the average rating that item I_j has received from all users that have rated it. It has been noted that item-based collaborative filtering offers some advantages over the user-based version, such as being more stable (a user's tastes may change over time, while an item remains constant) and more suitable for offline preprocessing in large applications [16].

To generate recommendations for a user U_j, a user-based collaborative filter builds a set consisting of the k nearest neighboring (i.e. most similar) users to the user being analyzed. The

item histories of the users in this set are used to predict the future behaviour of U_j. Let us denote the set of k users most similar to U_j as N_j. Jannach et al. [16] note that including users who have a negative correlation with each other in these sets has a negative impact on both performance and the accuracy of recommendation. Thus, we consider only positive similarities when using a metric that can return negative values (such as Pearson's correlation coefficient). The collaborative filter predicts U_j's rating for a new item $i \notin I_{Uj}$ as:

$$pred(U_j, i) = \bar{r}(U_j, I_{U_j}) + \frac{\sum_{x \in N_j} sim_{Jaccard}(U_j, x) \cdot (r(x,i) - \bar{r}(x, I_x))}{\sum_{x \in N_j} sim_{Jaccard}(U_j, x)} \quad (2.6)$$

Item-based collaborative filtering operates in much the same way. An item-item collaborative filter will construct a set of the most similar items for each item. To predict how a new user $u \notin U_{Ij}$ would rate item I_j, we use this set of similar items:

$$pred(I_j, u) = \bar{r}(u, I_u) + \frac{\sum_{x \in N_j} sim_{Jaccard}(I_j, x) \cdot (r(u,x) - \bar{r}(u, I_u))}{\sum_{x \in N_j} sim_{Jaccard}(I_j, x)} \quad (2.7)$$

We can form a list of top-n recommendations for a user from the items that yield the n highest values in Eq. (2.6) or (2.7).

2.2 Content-based recommendation

For the majority of the items in a recommendation problem, few ratings are available [33], causing the neighborhoods formed in collaborative filtering to be based on a small number of shared ratings. An alternative approach is to identify the attributes of each item, and directly

compare them to the user's preferences. By associating each item with a set of attributes that describe it, we can directly find similarities between items and users interests, without the need for users to link them by providing ratings [19], [27]. The main disadvantage of this type of approach is that data describing the content of each item is required. As such, being able to automatically generate a description of an item's content is highly desirable.

Historically, content-based recommendation has been applied to text-based items, where it is possible to extract a set of keywords from the document itself to provide a description. A well-known method for this is *term frequency/inverse document frequency (TF-IDF)* [16]. *Term frequency* indicates the rate at which a keyword *A* appears in a document *B*, and can be calculated in a normalized form to prevent longer documents from having universally higher term frequencies:

$$TF(A,B) = \frac{count(A,B)}{topCount(B)} \tag{2.8}$$

where *count(A,B)* indicates the number of occurrences of term *A* in document *B*, and *topCount(B)* is the number of occurrences of the most frequent term in document B. This measure is combined with the *inverse document frequency*, which reduces the weight of terms that are common between many documents, operating on the idea that very common terms are not helpful in distinguishing between items:

$$IDF(A) = \log\left(\frac{N}{count(A)}\right) \tag{2.9}$$

where N is the total number of items, and *count(A)* indicates the number of documents in which term A appears. Simply taking the product of the *TF* and *IDF* measures for each term present in a document generates a weighted vector which represents the most descriptive keywords for an item.

Once we have a content representation for each item, we can define the similarity between items in terms of the distance between their content vectors. For documents, this could be a Euclidean distance (or other distance metric) between TF-IDF vectors.

Analyzing the content of items for keywords may not always be feasible. The items present in the recommender system might not be books or articles, but rather restaurants or concert venues. Additionally, it may be time consuming the process the full text of a large amount of items for keywords, for example, for items in a library that do not have digital versions. Content-based approaches can still be applied in these cases by operating on a list of tags that define the content of the item. For books, a logical tagging scheme is the genres it belongs to (mystery, thriller, fantasy, romance, etc.). The Jaccard similarity index [30] can be used to find similarity between items A and B based on their tags:

$$sim(A,B) = \frac{|T_A \cap T_B|}{|T_A \cup T_B|} \tag{2.10}$$

T_A and T_B represent the sets of tags associated with items A and B, respectively. As with Eq. (2.3), the numerator is the number of genre tags shared between A and B, and the denominator is

the total number of genre tags assigned to *A* or *B*. A variety of content-based techniques exist to provide personalized recommendations to users. A simple approach, *k-nearest neighbors(kNN)*, recommends items which are most similar to those the user has previously chosen [16]. A range of other approaches, including Rocchio's algorithm, decision trees, linear classifiers, and probabilistic methods [19] have been applied to content-based recommendation systems. Agarwal et al. [34] propose a matrix factorization method which utilizes item meta-data to predict ratings. Their system learns topics associated with keywords on items, and subsequently predicts a user's rating for an item by determining the user's affinity to the topics of the item. This method has been shown to be effective in increasing prediction accuracy for sparse data, helping to alleviate the cold-start problem. Sieg et al. [35] study how content data can be utilized to improve collaborative recommendation. In their work, they collect meta-data tags for books from Amazon.com, and subsequently build user profiles based on the content data that underlies each user's item history. To generate the neighborhoods required for collaborative filtering, they use a similarity measure based on the level of interest for each user in each topic, rather than the classic method of using co-ratings. This content-based approach also proved useful in overcoming the cold-start problem.

2.3 Context-aware recommendation

In most recommendation applications, the user-item matrix tends to be sparse with few ratings available for the majority of the items present [33]. This can lead to links being formed between users based on a low number of co-rated items, even if their interests significantly differ,

yielding recommendations that are uninteresting to each other. To improve the quality of the recommendations, we consider the *context* of a user or item. As defined in [36]:

> "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the applications themselves."*

The task of a traditional recommender system is to establish a mapping from a set of users and items to ratings based on a set of past ratings [18], [20]. Context-aware recommendation systems expand on this approach, and consider contextual information as another dimension for the rating function. Contextual information can be incorporated into a recommendation system in a variety of ways [37], [38], [39]. For a recommendation engine that works with books, one definition of context is the availability of a book, or which bookstores or libraries currently have a copy in stock [19] [27]. In the same vein, a recommender dealing with movies may consider when certain theatres will show a movie and for what cost. Users may also choose movies based on the group of people they are with; parents with young children may be interested in completely different genres depending on whether or not their kids are present. Recommendations for a venue hosting concerts will be of interest to different clientele depending on the type of bands playing, the show times, and the price of admission. Restaurants may attract customers with different deals or special menus on particular days. Thus, explicit or implicit ratings assigned by a person can take on different interpretations depending on context, and a user's current context can potentially render much of their past item history largely irrelevant to the situation at hand.

3 Fuzzy Recommendation Systems for Public Library Catalogs

To see how fuzzy logic can be useful in designing a recommender, let us consider the meaning of a rating. Commonly expressed as "*k* stars out of 5 (or 10)," this rating is a measure of the rater's opinion about an item. Put another way, it is *the degree to which the rater favors the item*. It is thus natural to interpret a rating as the membership of that item in the fuzzy set "things this rater favors." One possible application of this interpretation is to predict ratings by treating known rankings as a fuzzy relation, and taking the *sup-T* composition between the relation and a new item [40]. Another way to interpret ratings data comes from the notion of a neighborhood in collaborative filtering; we can consider these neighborhoods to be fuzzy clusters within the item-item space, and use fuzzy clustering algorithms to discover them.

3.1 Related Work

A number of proposed recommenders follow the fuzzy clustering approach. Hino et al. [41] discuss the creation of a mixture of probability models to order items into tiered preference groups for each user. They utilize a modified version of fuzzy c-means [42] which determines the cluster count automatically to resolve the inputs from separate models. Treerattanapitak et al. [43] propose a modified approach to neighborhood generation for collaborative filtering, based on fuzzy c-means. The objective function of the fuzzy c-means algorithm is reformulated with an exponential term to give lower membership values to less relevant data points. Forsati et al. [44] propose a fuzzy clustering approach to recommend web pages to users, utilizing web content data as a set of keywords to define the similarity between web pages and generating user profiles

from their browsing history. Their clustering approach is a variant of fuzzy c-means based on genetic k-means clustering [45] [46].

A variety of other fuzzy recommendation algorithms have also been explored. Bobadilla et al. [47] discuss the use of genetic algorithms to improve on the results of basic collaborative filtering. They add a weight vector to the similarity function between users that is learned by a genetic algorithm whose objective function is the mean absolute error of the recommender system. Capuruço et al. [48] study a user-user fuzzy trust metric. They apply two fuzzy inputs: "degree of homophily" (low/medium/high) and "degree of separation" (close/medium/far) between users and use a set of decision rules to determine a level of trustworthiness based on these fuzzy inputs. Wu et al. [49] discuss a recommendation algorithm for telecom products employing a hybrid of item-based and user-based collaborative filtering. They propose a similarity measure based on trees [50] with fuzzy numbers applied to model uncertain weights and linguistic variables. Cornelis et al. [51], [52] have applied fuzzy logic and a content-based approach hybridized with collaborative filtering to produce recommendations in the domain of one-and-only recommendations. In this domain, traditional collaborative filtering is ineffective as it can only provide recommendations for items that have been purchased in the past, which is meaningless for events which are not recurring. Porcel et al. [53] discuss how fuzzy linguistic modeling can be applied to recommendation. In particular, they study how to provide recommendations in the context of technology transfer offices, recommending appropriate resources to researchers in their areas of interest.

Many related works survey how recommender systems can be applied in the domain of *University Digital Libraries.* Porcel et al. study how fuzzy linguistic modeling [54] can be used to recommend relevant articles from the research areas of the system's users; in addition, they review how problems caused by the ever-increasing amount of information can be managed [55]. Morales-del-Castillo et al. [56] propose a multi-agent model which considers the value of recommending the latest acquisitions to keep researchers on the cutting edge. Matsatsinis et al. [57] investigate integrating the user in the recommendation process for scientific papers. Huang et al. [58] study hybridizing collaborative filtering and content-based approaches by employing a Hopfield network. Herlocker et al. [59] perform live experiments with collaborative filtering for digital libraries, and find that they can increase the quality of the search interface, improving the user experience by reducing the amount of times that queries need to be reformulated. In general, these investigations focus on research libraries with much larger holdings and a much more engaged user base than a rural library. Consider a typical research library in a university; one normally expects tenured professors to maintain their research interests over many cycles of student recruitment, study and graduation. Thus their new students will follow discovery paths through the library that strongly resemble each other and those of previous students (i.e. they are a small, tightly-focused gestalt). In this situation, a recommender should be exceptionally effective. In the broader domain of digital libraries (e.g. the ACM or IEEE libraries), a similar situation holds; researchers tend to be part of communities, and the resources that one community member finds useful are likely also useful to others, meaning that the research community acts as a gestalt. Again, this is a situation where recommenders should excel. In particular, the problem of sparsity can be expected to be less important in the digital/research library context compared to the rural library context.

3.2 Hybrid Fuzzy Recommender

This section describes a content-based fuzzy recommendation method and discusses its advantages and disadvantages with respect to traditional collaborative filtering. Additionally, our method for hybridizing this algorithm with item-item collaborative filtering is introduced.

3.2.1 Fuzzy taste vector recommendation

The concept of using a fuzzy model describing the relative importance of features to a user to generate recommendations [22] can be adapted to the problem at hand. Fuzzy sets supply a means with which we can model a user's degree of preference for particular item features. A fuzzy set S in the universe of discourse X can be described by a set of pairs: $\{(x, \mu_s(x))$ for x in $X\}$ where $\mu_s(x)$ represents the membership of value of attribute x in the set S [60]. The membership value can be interpreted as the degree of preference a user has for a particular attribute. A high membership value for a particular attribute reflects a user's desire to select items which possess that attribute.

We use the item history of each user to generate a fuzzy taste vector which indicates their preferences. Given a set of content tags associated with an item, we assign a value to each tag that describes how a user would rate an item having that tag. The fuzzy taste vector for an item is then the concatenation of each tag valuation. Books, for example, can be described by their

genre (mystery, suspense, thriller, fantasy, etc.), author, etc. The membership value μ_U the user U assigns to each tag T for which they have provided at least one rating is calculated as:

$$\mu_U(T) = \frac{\sum_{I \in R(U,T)} rating(U,I)}{|R(U,T)|} \tag{3.1}$$

where $R(U,T)$ is the set of items rated by user U with tag T, and $rating(U,I)$ is the rating user U assigns to item I, normalized to the range [0,1]. If a user has not rated any items with tag T, the membership value is not calculated, as we have no indication of their level of preference for items of that genre.

This vector can then be used to predict a rating for a new item according to:

$$pred(U,I) = \frac{\sum_{t \in T(I) \cap T(U)} \mu_U(t)}{|T(I) \cap T(U)|} \tag{3.2}$$

where $T(I)$ is the set of tags associated with item I, and $T(U)$ is the set of tags associated with all items user U has previously rated. This equation is only applied in cases where there is some overlap between $T(I)$ and $T(U)$. In the event that none of the tags overlap, meaning that a user has never rated an item from any of the genres associated with item I, we cannot generate a meaningful prediction for that user-item pair. Algorithm 3 describes how taste vectors are generated for each user in our recommendation system:

This approach provides some key advantages over collaborative filtering. First, generating a user's taste vector is done in linear time with respect to the amount of ratings they have provided. In collaborative filtering, generating a neighborhood for a user is more computationally expensive, requiring *O(nm)* time where *n* is the user's past ratings, and *m* is the number of users who have also given ratings on the *n* items. Additionally, as soon as a user has rated one item from a genre, this method can generate a prediction for any item in that genre, rather than limiting predictions to items rated by neighboring users/items. However, this method relies on the presence of meaningful and detailed content data. The effects of differing quality in the content data on this approach are investigated in 3.4.1.

3.2.2 Hybridization

Hybridizing predictions from different recommendation algorithms can help overcome the shortcomings of individual approaches [16]. Our approach applies a hybrid of item-item collaborative filtering (Eq. (2.7)) and fuzzy taste vector recommendation (Eq. (3.2)). We first define a measure of prediction confidence for each of these two approaches. The confidence value for a prediction from item-item collaborative filtering is defined by the total similarity between the item whose rating is being predicted and the items previously rated by the user:

$$confidence(U, I) = \sum_{B \in (N \cap R(U))} sim(I, B) \tag{3.3}$$

where N represents the set of items in the neighborhood of item I, and $R(U)$ represents the set of books rated by user U.

For a fuzzy taste vector recommendation, we define the confidence value to be the count of tags that overlap between an item and the user's taste vector:

$$confidence(U, I) = \left| T(I) \cap T(U) \right| \tag{3.4}$$

Where $T(I)$ is the set of tags for an item, and $T(U)$ is the set of tags in a user's taste vector (note that these are crisp sets and the standard intersection). We expect that higher confidence values imply the predictions we can make for a user's rating on an item will be more accurate.

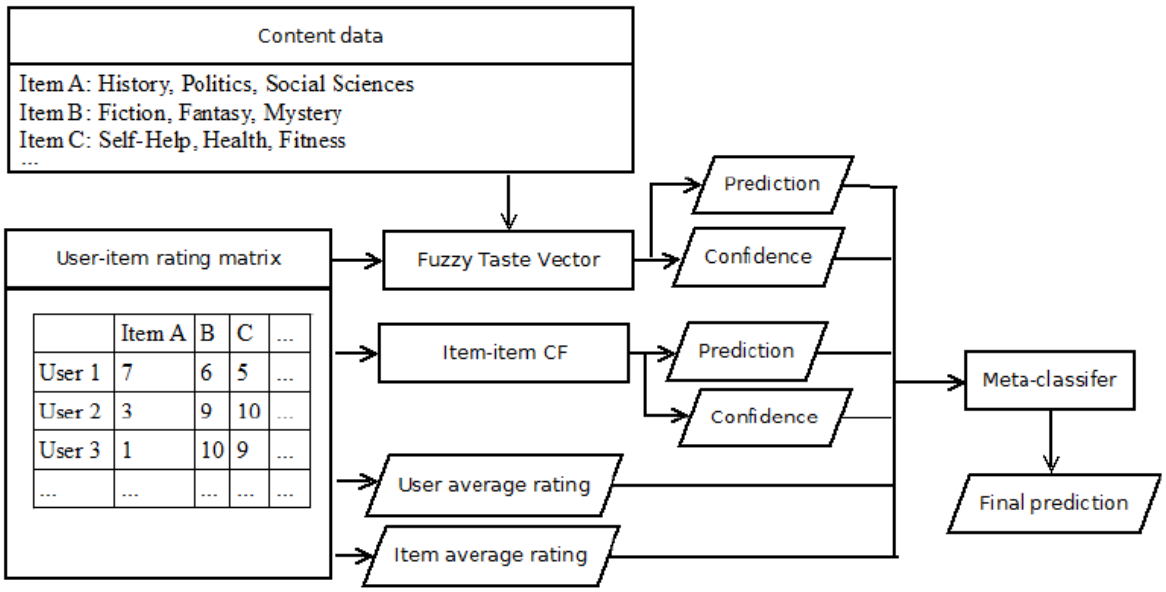

Figure 3.1: Classifier Ensemble

At this point, we have generated the item-item collaborative filtering prediction (Eq. (2.7)) and confidence level (Eq. (3.3)), the fuzzy taste vector prediction (Eq. (3.2)) and confidence (Eq. (3.4)). We also include the user's average rating over all past items and the item's average rating to give a final set of 6 attributes. We then train a new classifier to fuse these six attributes into a

final predicted rating; thus, our overall algorithm can be seen as a classifier ensemble. This full design is captured in Figure 3.1: Classifier Ensemble. The final prediction is calculated by a classifier algorithm trained to determine a user's actual rating for an item based on the 6 attributes. The classifiers we use in our experiments include k-nearest neighbors, radial basis function networks, multilayer perceptrons, linear regression, and decision tables. The final prediction is calculated by a classifier algorithm trained to determine a user's actual rating for an item based on the 6 attributes. The classifiers we use in our experiments include k-nearest neighbors, radial basis function networks, multilayer perceptrons, linear regression, and decision tables.

For an example of how the system operates, consider the user-item matrix and genre information provided in Table 3.1 and Table 3.2.

Table 3.1: Example user-item rating matrix

	Book A	Book B	Book C	Book D
User 1	0.5	0.6	-	-
User 2	0.7	0.8	0.7	0.8
User 3	-	0.7	0.6	0.7
User 4	-	0.7	0.6	**?**

Table 3.2: Example genre information

Book	Genres
A	Fiction, Fantasy, War
B	Fiction, Science fiction, Romance
C	Fiction, Horror, Romance
D	Non-fiction, History, War, Romance

To predict the rating for user 4 on item D, we first calculate the similarity between each item according to Eq. (2.3) and (2.5). For the similarity between book A and B, we find:

$$sim(A,B) = \frac{(0.5-0.55)\cdot(0.6-0.55)+(0.7-0.75)\cdot(0.8-0.75)}{\sqrt{((0.5-0.55)^2+((0.7-0.75)^2)\cdot((0.6-0.55)^2+(0.8-0.75)^2)}} = -1$$

from Eq. (2.5), and $sim_{Jaccard}(A,B) = sim(A,B) \cdot \frac{2}{4} = -0.5$ from Eq. (2.3). Note that this negative value indicates that these books are actually dissimilar, as both users that provided ratings for them rated A lower than average and B higher than average. The remainder of book similarities are shown in Table 3.3.

Table 3.3: Item-item similarity values

	Book A	Book B	Book C	Book D
Book A	-	-0.5000	0.2500	-0.3333
Book B	-0.5000	-	-0.7115	0.5000
Book C	0.2500	-0.7115	-	-0.4703
Book D	-0.3333	0.5000	-0.4703	-

In a large-scale implementation of this algorithm, a maximum neighborhood size would be selected to reduce calculation time and ignore the input from more distant neighbors of an item. For this small-scale example, each book only has one neighbor with a positive correlation: book A and C are neighbors, as are book B and D.

The next step in our system is to calculate the fuzzy taste vector for each user, as determined by Eq. (3.1). To find user 2's value for fiction, we consider their rating for the books they have rated which have that genre (in this case, book A, B, and C). We calculate the value by $\mu_U(T) = \sum_{I \in R(U,T)} \frac{rating(U,I)}{|R(U,T)|} = \frac{0.7+0.8+0.7}{3} = 0.7333$. The values for all users across each genre are

presented in Table 3.4, with blank fields indicating that the user has never rated an item with that particular genre.

Table 3.4: Calculated fuzzy taste vector values

User	Fantasy	Fiction	History	Horror	Non-fiction	Romance	Science fiction	War
1	0.5000	0.5500	-	-	-	0.6000	0.6000	0.5000
2	0.7000	0.7333	0.8000	0.7000	0.8000	0.7667	0.8000	0.7500
3	-	0.6500	0.7000	0.6000	0.7000	0.6667	0.7000	0.7000
4	-	0.6500	-	0.6000	-	0.6500	0.7000	-

With item-item similarities and the fuzzy taste vectors calculated, we can generate predictions for the target value (User 4 - Book D) using Eq. (2.7) and (3.2). Additionally, we can find the confidence values for each prediction from Eq. (3.3) and (3.4). For user 4's rating on book D, we find the item-item collaborative filtering prediction to be $pred(4,D) = \bar{r}(4) + \frac{sim(D,B)\cdot(r(4,B)-\bar{r}(4))+}{sim(D,B)} = 0.65 + \frac{0.5(0.7-0.65)}{0.5} = 0.6$ from Eq. (2.7). The fuzzy taste vector-based prediction is given by $pred(4,D) = \sum_{t\in T(4)\cap T(D)} \frac{\mu_4(t)}{|T(4)\cap T(D)|} = \frac{0.6500}{1} = 0.6500$ (Eq. (3.2)). We find the item-item collaborative filtering confidence value to be 0.5 from Eq. (3.3), and the fuzzy taste vector confidence to be 1.0 from Eq. (3.4). Finally, we calculate user 4's average rating, 0.65, and book D's average rating, 0.75.

To determine the final prediction, the values generated thus far are fed into a classifier algorithm that has learned from training data. In our experiments, we use ten-fold cross validation to generate predictions from each part of the hybrid engine for each instance. For simplicity in this example, we generate training instances for the classifier by predicting some existing ratings in

the problem (user 2-book D, user 3-book D, and user 3-book B). The input to the final classifier is shown in Table 3.5.

Table 3.5: Prediction example

Fuzzy taste vector		Collaborative filtering		Average rating		Actual Rating
Prediction	Confidence	Prediction	Confidence	User	Item	
0.6500	**1.0000**	**0.6000**	**0.5000**	**0.6500**	**0.7500**	**?**
0.7792	4.0000	0.8000	0.5000	0.7500	0.7500	0.8
0.6917	4.0000	0.7000	0.5000	0.6667	0.7500	0.7
0.7667	3.0000	0.7000	0.5000	0.6667	0.7000	0.7

To predict the actual rating for the test instance, a k-nearest neighbors classifier would find the mean actual rating from its closest neighbors in the training set. For the values presented in Table 5, given that the last 3 rows are the nearest neighbors to the first instance, a 3-nearest neighbors classifier would predict the unknown actual rating as $\frac{0.8+0.7+0.7}{3}$, generating a final prediction of 0.7333.

3.3 Experimental Methodology

This section describes the details of the datasets which we have selected for experimentation, as well as a discussion of evaluation metrics.

3.3.1 Datasets

The Book-Crossing dataset [23] contains 278,858 users providing 1,149,780 ratings for 271,379 books. 433,671 ratings are explicit (given on a scale of 1-10), and the remaining 716,109 ratings

are implicit. For the experiments considered here, only the set of explicit ratings is used. The sparsity (Eq. (2.1)) of this dataset is 99.99%.

To augment the rating data provided in BookCrossing, additional content data was obtained from Amazon Web Services [61] for 230,347 of these books, consisting of a classification tree that assigns 3133 content tags which describe the genre of a book (for example: history, anthropology, psychology, fiction, mystery, etc.). The average number of tags per book is 19, with a minimum of 3 and a maximum of 153. A second set of content data was obtained from the (MARC) records held at the Library of Congress [62], which provides tags for 57,695 books, consisting of 19,450 unique subject tags with an average of 3 tags per book, a minimum tag count of 1 and a maximum of 19. Experiments with these sets of content data are of course limited to the set of books for which the content data exists.

The MovieLens dataset [24] contains ratings on a scale of 1-5 for a selection of movies. Three different subsets are available:

- MovieLens 10M, consisting of 10 million ratings on 10,000 movies by 72,000 users (98.6% sparsity)
- MovieLens 1M, consisting of 1 million ratings on 4000 movies by 6000 users (95.8% sparsity)
- MovieLens 100k, consisting of 100,000 ratings on 1682 movies by 943 users (93.7% sparsity)

These datasets also include content data consisting of 19 unique tags, with an average of 2 tags associated with each movie, a minimum of 1, and a maximum of 8. For the experimental results we present, we use the MovieLens 100k subset to facilitate comparison with other works.

3.3.2 Experimental setup and evaluation

All of the results we present in this chapter are obtained from performing ten-fold cross validation experiments [63]. We select two popular measures as evaluation metrics to facilitate comparisons with other works: *Mean Absolute Error (MAE)* and *Root Mean Squared Error (RMSE)*. *MAE* is defined by:

$$MAE = \frac{\sum_{r \in R} |prediction(r) - actual(r)|}{|R|} \tag{3.5}$$

where R represents the set of ratings to be predicted. *RMSE* is defined as:

$$RMSE = \sqrt{\frac{\sum_{r \in R} (prediction(r) - actual(r))^2}{|R|}} \tag{3.6}$$

Prediction coverage [64] is another metric used to demonstrate the quality of recommendation algorithms. A recommendation algorithm may be unable to make a personalized prediction for certain users on some items; this occurs in collaborative filtering when the item to be predicted is not in the neighborhoods of any item the user has previously rated. If an algorithm is highly accurate but can only predict a small portion of the items available, it may not be as desirable as an algorithm which has less accuracy but can predict the rating for any item. Plainly, in sparse datasets such as we expect to encounter in rural libraries, prediction coverage becomes a

critically important consideration. *Prediction coverage* for both collaborative filtering and fuzzy taste vector recommendation can be defined as follows:

$$\text{coverage} = \frac{\sum_{u \in U_T} \left\| \{ i \mid i \in T(u) \wedge confidence(u,i) > 0 \} \right\|}{\sum_{u \in U_T} \left\| T(u) \right\|} \tag{3.7}$$

where U_T is the set of all users who have ratings in the test set, *T(u)* is the set of all items for which user *u* has a rating in the test set, and *confidence(u,i)* is the confidence value for the prediction output from the collaborative filter or fuzzy taste vector, as defined by Eq. (3.3) and (3.4). A confidence value of 0 indicates that the recommender has no data to relate a particular user to a particular item, and thus cannot generate a meaningful prediction for it.

3.4 Experimental Results

This section provides experimental results showing the improvement from hybridizing item-item collaborative filtering with a fuzzy taste vector recommendation; and, compares our results with papers studying the same datasets.

3.4.1 System Design

We first consider the effect that different similarity metrics have on the quality of our overall results. To this end, we compare the mean absolute error of item-item collaborative filters using adjusted cosine weighted with the Jaccard coefficient and cosine similarity in

Table 3.6.

Table 3.6: Similarity metric comparison

Dataset	Neighborhood size	MAE	
		Adjusted Cosine	Cosine
Book-Crossing	3	0.819	0.863
	10	0.961	1.017
	20	1.023	1.069
	30	1.056	1.108
	50	1.094	1.148
	100	1.148	1.194
MovieLens	3	0.758	0.796
	10	0.745	0.775
	20	0.732	0.762
	30	0.726	0.754
	50	0.719	0.744
	100	0.716	0.737

From these results, it is clear the adjusted cosine similarity offers superior performance on these datasets, and thus, it is used in further experiments. In Table 3.7, we explore the accuracy and coverage of an item-item collaborative filter in greater detail.

Table 3.7: Item-item collaborative filtering

Dataset	Neighborhood size	MAE	RMSE	Coverage
Book-Crossing	3	0.819	1.432	0.068
	10	0.961	1.561	0.130
	20	1.023	1.606	0.179
	30	1.056	1.633	0.213
	50	1.094	1.661	0.262
	100	1.148	1.709	0.339
MovieLens	3	0.758	1.064	0.654

	10	0.745	1.002	0.865
	20	0.732	0.968	0.929
	30	0.726	0.952	0.952
	50	0.719	0.936	0.972
	100	0.716	0.923	0.987

For Book Crossing, we observe a clear tradeoff between coverage and accuracy in Table 3.7 as the neighborhood size increases. On the less sparse MovieLens data, we find that our prediction accuracy increases alongside coverage as we increase the neighborhood size. This finding is, obviously, crucial to domains with sparse data; the collaborative filter by itself is unlikely to give adequate performance in such sparse datasets. The question is, does the fuzzy taste vector improve these results? We explore this question in Table 3.8.

Table 3.8: Fuzzy taste vector recommendation

Dataset	Neighborhood size	MAE	RMSE	Coverage
Book Crossing – Marc data	-	1.286	1.796	0.619
Book Crossing – Amazon data	-	1.264	1.685	1.000
Book Crossing – Amazon data	3	0.978	1.326	0.068
	10	1.022	1.382	0.130
	20	1.046	1.414	0.179
	30	1.059	1.431	0.213
	50	1.069	1.440	0.262
	100	1.088	1.459	0.339
MovieLens	-	0.828	1.049	0.991
MovieLens	3	0.798	0.999	0.654
	10	0.812	1.022	0.865
	20	0.819	1.032	0.929
	30	0.822	1.038	0.952
	50	0.824	1.043	0.972
	100	0.827	1.047	0.987

In Table 3.8, we compare the results of the fuzzy taste vector with or without neighborhoods. Fuzzy taste vector recommendation does not require a neighborhood size, and these basic results are given in the first two rows of the Book Crossing results, and the first row of the MovieLens results. However, this provides an incomplete picture, as the collaborative filter only provided predictions on some of the items. In order to make a complete comparison, we have also included the results from making predictions for only the items that were possible for the item-item collaborative filtering algorithm to predict with a given neighborhood size (listed in column 2 of Table 3.8). We note that the Book Crossing experiments using the data collected from Amazon [61] result in better accuracy and coverage than those using MARC records from the Library of Congress [62]. For this reason, our further experiments focus on using the Amazon data. Additionally, we find that applying content data is helpful in overcoming the sparsity problem faced in the Book Crossing dataset, providing better prediction coverage than collaborative filtering, as well as lower prediction error for neighborhoods larger than 30. However, for the smaller neighborhood sizes (where the collaborative filter was most accurate), the fuzzy taste vector was less accurate. The picture is quite different for MovieLens, as the relatively greater density of the data lends itself well to collaborative filtering. The content data available for the movies is furthermore less detailed than the content data collected from Amazon for Book Crossing, and so the fuzzy taste vector was inferior to collaborative filtering on this dataset.

Thus, what we find is that the two individual classifiers offer different trade-offs between accuracy and coverage, with the differences being accentuated as the sparsity of the dataset

increases. We now proceed to explore the full design described in 3.2.2, by selecting and parameterizing a meta-classifier that utilizes the six attributes identified in 3.2.2.

Table 3.9: Meta-classifier experiments (neighborhood size = 100)

Dataset	Meta-classifier	MAE	RMSE	Coverage
Book Crossing	**K-nearest neighbors**	**1.045**	**1.430**	**0.339**
	Radial basis function network	1.062	1.443	0.339
	Multilayer perceptron	1.194	1.523	0.339
	Linear regression	1.077	1.461	0.339
	Decision table	1.062	1.455	0.339
MovieLens	**K-nearest neighbors**	**0.712**	**0.907**	**0.987**
	Radial basis function network	0.739	0.922	0.987
	Multilayer perceptron	0.740	0.927	0.987
	Linear regression	0.713	0.908	0.987
	Decision table	0.718	0.910	0.987

In Table 3.9 we present our results for five different meta-classifiers (we only present out-of-sample results for the best parameterization found for each classifier). A simple k-nearest-neighbors approach returned the best results for both datasets. On the MovieLens dataset, the ensemble classifier was more accurate than the base algorithms for any neighborhood size, and the prediction coverage was equal. On the BookCrossing dataset, the result is more nuanced; at a neighborhood size of 100 (chosen for maximum coverage), the coverage is as good as the fuzzy taste vector or the collaborative filter, and more accurate than either. As previously noted the collaborative filter was more accurate at smaller neighborhood sizes on this dataset; while the fuzzy taste vector *without a neighborhood size restriction* had better prediction coverage. To compare the quality of the meta-classifier and the fuzzy taste vector method at the same level of

prediction coverage, we augment the predictions made by the meta-classifier with predictions from the fuzzy taste method alone for any ratings that the meta-classifier could not predict. The result of this is shown in Table 3.10.

Table 3.10: Meta-classifier with fuzzy taste predictions outside neighborhood

Dataset	Meta-classifier	MAE	RMSE	Coverage
Book Crossing	K-nearest neighbors	1.189	1.603	1.000

Plainly, this design improves prediction coverage but at the cost of reduced accuracy. After considering these results, we determined that the simple kNN meta-classifier of Table 3.9 was the best trade-off of accuracy and coverage for these two datasets.

3.4.2 Comparison

Hino et al. [41] evaluate their approach on a subset of the Book-Crossing dataset which ignores users with less than 8 ratings and items with less than 12 ratings. For a direct comparison with our algorithm, we have recreated this subset.

Table 3.11: Book-Crossing comparison

Setup	MAE	Coverage
Hino et al. [41]	1.253	-
KNN hybrid (neigh only)	1.015	0.262
KNN hybrid (fuzzy outside)	1.139	1.000

In addition to [41], we also compare our Book Crossing results against two methods that do not use fuzzy logic. In the first of these, Agarwal et al. [34] experiment on a different subset of the

Book-Crossing dataset, where any rating less than 5 on the 1-10 scale is removed. Once again, we have recreated this dataset for a direct comparison against our system in Table 3.12.

Table 3.12: Book-Crossing comparison

Setup	MAE	Coverage
Agarwal et al. [19]	1.032	-
KNN hybrid (neigh only)	0.953	0.309
KNN hybrid (fuzzy outside)	1.046	1.000

Unfortunately, coverage is not discussed by Agarwal et al., so we cannot claim that our approach is more accurate in all cases. We do find, however, that we achieve better prediction accuracy within the collaborative filtering neighborhood, and slightly worse accuracy when incorporating fuzzy taste vector predictions for the full dataset.

Sieg et al. [35] present results on yet another subset of the full Book-Crossing data, removing all users with fewer than 20 ratings to reduce the sparsity of the dataset. Also, as part of the method in this work, the authors have collected content data from Amazon for 75,646 books (27.9% of the dataset), and limit their experiments to this subset. Unfortunately, this prevents us from creating the same subset as the authors of this paper, but we attempt to approximate it by removing users with fewer than 20 ratings from our set of 230,347 books with content data. Thus, our comparisons in Table 3.13 are imprecise, although we believe that they remain fairly accurate. We find that our method yields superior prediction coverage to [35], though it is at the cost of greater error. Restricting the scope of our predictions to the instances for which the collaborative filter can make a prediction results in a lower error, but less coverage. To give

more points of comparison, we also include values in Table 3.13 resulting from restricting our predictions to those above certain confidence thresholds as determined by Eq. (3.4). This does yield a point at which we find slightly less error and better coverage, however, this difference is likely too small to be reliable given the approximate nature of our reconstruction. The added benefit of our method is the ability to choose between accurate predictions on a more limited set of items and less accurate predictions with the potential to recommend a more diverse array of items.

Table 3.13: Book-Crossing comparison

Setup	MAE	Coverage
Sieg et al. [20]	1.025	0.506
KNN hybrid (fuzzy outside)	1.086	1.000
KNN hybrid (confidence restricted)	1.065	0.762
	1.029	0.653
	1.021	0.534
	1.010	0.429
KNN hybrid (neigh only)	0.997	0.412

Most of the papers surveyed in 3.1 provide experimental results from their algorithms on the MovieLens 100k subset. Table 3.14 provides their best results in terms of MAE, compared with the top result from our hybrid method.

Table 3.14: MovieLens comparison - MovieLens 100k data

Setup	MAE	Coverage
Hino et al. [41]	0.712	-
Treerattanapitak et al. [43]	0.765	-
Kwon et al. [65]	0.790	-

Capuruço et al. [48]	0.750	-
Zenebe et al. [60]	0.780	-
Bobadilla et al. [47]	0.750	0.850
KNN hybrid	0.712	0.987

3.5 Summary

In this chapter, we have investigated the development of a hybrid collaborative / content filter recommendation engine for integrated library systems. The key expected characteristic for this domain is extreme sparsity, and so maximizing both predictive accuracy and predictive coverage are essential goals. We have examined the impact that various neighborhood sizes have on accuracy and coverage on two datasets with different levels of sparsity. We proposed an ensemble classifier to combine the results of a basic nearest-neighbor collaborative filter with a content filter based on "fuzzy taste vectors". Our proposed algorithm is at least competitive with all existing fuzzy recommenders on these datasets. Our meta-classifier is as accurate as any other on the MovieLens data, and the most accurate on the BookCrossing data. While we did find that we could substantially increase accuracy on BookCrossing by using a non-fuzzy collaborative filter, this comes at the price of very poor prediction coverage. Utilizing confidence levels to restrict the set of predictions our system makes enables us to find a balance between producing accurate recommendations and providing better coverage of the items in the system. The confidence level threshold can be adjusted depending on the user's preference for highly relevant recommendations versus more diverse recommendations.

4 Context-Aware Recommendation in Location-Based Social Networks

4.1 Related Work

Location-based social networks present a unique environment for a recommendation problem that differs from the classic recommendation task based on a catalogue of products [66]. When perusing a list of products from an e-commerce site, a user could choose to purchase any item that piques their interest, and thus, any item could potentially be recommended. For recommendations in location-based social networks to make sense, the user must be able to physically access a location. Furthermore, traditional collaborative filtering algorithms are designed for a single interaction between a user and item, while in an LSBN users are likely to revisit locations such as restaurants or coffee shops.

Gao et al. [7] [67] [68] propose a variety of approaches for generating recommendations for this domain. In [7], a geo-social correlation model (gSCorr) is proposed. This model captures the relationship between social and spatial factors by defining four geo-social circles: local friends, local non-friends, distant friends, and distant non-friends. The authors consider local users as those who live in the same state/province as each other, while distant users live in separate states/provinces. According to [26], friends that live 1,000 km apart from each other can have a more pronounced influence on the check-in behaviour of each other than local friends, providing the motivation for incorporating both local and distant users. The probability of a user visiting a location is then based on the similarity of that user to other users who have visited that location. The contribution of users from each geo-social circle to the final probability is weighted by parameters to be learned by the projected gradient method from training data.

The approaches studied in [67] and [68] focus on the benefits of considering temporal effects on user behaviour. In [67], Gao et al. propose a location recommendation framework based on dividing a user's check-in history into a set of temporal states, for example, a day could be broken up into 24 temporal states -- one for each hour. The *temporal consecutiveness property* implies that the check-in tendencies of users are likely to be similar in consecutive temporal states. A user could check-in at a location for lunch between noon and 1:00pm, or 1:00pm and 2:00pm, but as we move through more temporal states, the "lunch" check-in becomes less likely. This property is leveraged to construct a model of user behaviour. In [68], the authors employ the *Gaussian mixture model* to capture the probability distribution for a user's check-in tendencies at a location. This model allows for multiple centers at the temporal states that see greater check-in frequencies than surrounding states. Probability distributions are built for daily, weekly, and a combination of daily-weekly temporal states. These are then incorporated into a set of spatial models: the *Most Frequent Check-in Model,* the *Order-1 Markov Model* [69], and the *Social Historical Model* [70].

Wang et al. [71] focus on the factors that trigger a user to visit a location for the first time. By analyzing data collected from two real LBSNs, Brightkite and Gowalla, they conclude that the primary factors involved in a new visit are social (a user being encouraged to try a new location with their friends) and geographical (locations in the proximity of a user are better candidates for a new visit). To apply these theories, we propose and implement two algorithms: the *friendship-based bookmark-coloring algorithm (FBCA)* and the *location-friendship bookmark-coloring algorithm (LFBCA)*. FBCA involves calculating a *personalized page rank (PPR)* [72] for each

user, then distributing this value to the locations visited by each user. Then, locations geographically distant from the user are filtered out, and the locations that collect the highest values of PPR are recommended. LFBCA functions in a similar way, but first weights the similarity of users according to how often they have visited common locations in the past. The performance of each algorithm is evaluated on the data collected from Brightkite and Gowalla.

Ye et al. [10] [73] explore the influence of geographical and temporal factors for location recommendation. In [10], the phenomenon of geographical clustering is investigated and fused with user preferences and social influence, resulting in a probability model that predicts the most likely points of interest for each user. Temporal characteristics of points of interest are discussed [73], which examines how check-ins follow an hour-of-the-day/day-of-the-week cycle for different categories of locations. Ye et al. strive to identify feature types for locations based upon these check-in patterns to label locations such as a *Bar* or *College*, and discuss how these tags, once discovered, can be employed in future recommendations. Bao et al. [74] investigate how recommendations can be generated for users who are visiting new cities and do not necessarily have any shared check-ins with local users. This is accomplished by using categorical data associated with venues to learn a user's preferences in terms of tags such as *Italian food* or *Nightlife*, then utilizing this profile to find similar users in a different location. Their approach is evaluated on data collected from Foursquare, specifically for users that live in and/or travel between New York City and Los Angeles.

4.2 Spatial-temporal-social integrated recommendation

This section describes our approach to improve recommendation quality by incorporating contextual data into a location-based recommendation problem. We begin by assuming that locations are similar to items for a collaborative filter; that is, there is a set of possible location L that a user may check into, and we can determine the set $L_{Uj} \subseteq L$ of locations that user U_j has previously visited (we will henceforth refer to each visit to a location as a "checkin"). We assume that there may be repeated checkins at a single location, as users tend to visit certain locations on a regular or semi-regular basis. We denote the set of checkins by user U_j at location L_m by $C(U_j, L_m)$, and we adopt the convention that L_i, L_{i+1} denote chronologically consecutive checkins, with L_{i+1} the more recent. We integrate several types of contextual data relevant to location-based social networks into the collaborative filter; intuitively, the more data that is available to represent a user, the more accurately we can predict their interests.

Time is one type of contextual data that can be applied to recommendations. A user's interests can shift over time, rendering recommendations based on locations they visited months or years in the past less relevant than recommendations based on their recent activity [75] [76]. Additionally, a user may make multiple visits to a location over time, indicating a stronger preference for that location than a user who only visits once. To apply this concept, we calculate a value representing each user's current level of preference for every location they visited from the dates of their visits:

$$V(U_j, L_m) = \frac{\sum_{l \in C(U_j, L_m)} date(l) - date_s}{(date_t - date_s) \cdot f_{\max}(U_j)} \tag{4.1}$$

where $date(l)$ is the date of checkin $l \in C(U_j, L_m)$, $date_T$ is the current date, and $date_S$ is an arbitrary start date indicating the earliest check-in that will be considered. To keep this value in the range [0,1] for all users, the result of the sum is divided by $f_{max}(U_j)$, a function which returns the number of visits a user has made to their most frequently visited location over the time window from $date_S$ to $date_T$. This equation produces higher values for locations that a user has visited more frequently and more recently. In the event that a user stops visiting certain locations, this equation speeds up the rate at which a collaborative filter "forgets" about their old interests [15] [75], focusing future recommendations on locations the user is more likely to find relevant. Using the value calculated by Eq. (4.1) in place of ratings in Eq. (2.6), we calculate predictions as follows:

$$pred_{temporal}(U_j, i) = \overline{V}(U_j, I_{U_j}) + \frac{\sum_{x \in N_j} sim_{Jaccard}(U_j, x) \cdot (V(x,i) - \overline{V}(x, I_x))}{\sum_{x \in N_j} sim_{Jaccard}(U_j, x)} \quad (4.2)$$

Social networking data between users is another type of context which we incorporate into the recommendation algorithm. Traditional user-user collaborative filtering determines the similarity between users based entirely on the similarity of their item histories [15] [16]; this can be expanded when social data directly linking pairs of users is available [7] [74]. Collaborative filtering relies on the pattern where a pair of users who have visited a similar set of locations will tend to visit a similar set of locations in the future, whether or not they are aware of each other. This does not account for the possibility that a user may directly influence the choice of another. To take into consideration the case where a user recommends a location to their friends, or a group of friends attend a location as a group, explicit social ties can be incorporated into the similarity measure from Eq. (2.3). Our approach to this is to simply decrease the similarity between users who are not friends:

$$sim_{social}(U_j, U_k) = \begin{cases} sim(U_j, U_k) \cdot \left(\alpha + \omega_{jk}(1-\alpha)\right) & U_k \in F_j \\ sim(U_j, U_k) \cdot \alpha, & U_k \notin F_j \end{cases} \quad (4.3)$$

$sim(U_1, U_2)$ is the similarity value between users U_1 and U_2, F_1 is the set of users who are friends with U_1, and α is a constant in the range [0,1]. Lower values of α indicate that a greater importance is placed on users who are friends, with a value of 0 meaning that users who are not friends have no impact on each other's recommendations, and a value of 1 meaning that there is no difference in the weight placed on friends and non-friends. ω_{12} is a weight factor which represents the strength of the friendship between users U_1 and U_2. We find a value for ω as follows:

$$\omega_{jk} = \frac{\left|C(U_{j'}*) \cap C(U_{k'}*)\right|}{\left|C(U_{k'}*) \cap C(U_{k'}*)\right|} \quad (4.4)$$

where $C(U_j,*)$ is the set of all check-ins made by user U_j at any location. We treat check-ins by two users as intersecting, if they are made at the same location within one hour of each other, implying that the users may have visited that location as a group. This weighting factor adopts the paradigm that if two users visit a location together, they share a greater similarity than users who visit a location separately, and will have a more pronounced influence on each other's future behaviour [16]. Predictions output by the social recommender are calculated similar to Eq. (2.6) with the similarity between users determined by Eq. (4.4):

$$pred_{social}(U_j, i) = \bar{r}(U_j, I_{U_j}) + \frac{\sum_{x \in N_j} sim_{social}(U_j, x) \cdot \left(r(x,i) - \bar{r}(x, I_x)\right)}{\sum_{x \in N_j} sim_{social}(U_j, x)} \quad (4.5)$$

The physical location of a user has significant implications on which locations they will visit [66]. Unlike an e-commerce or library recommendation problem where a user has access to all

items, a user must be able to travel to a location. To some extent, collaborative filtering naturally restricts location recommendations to those in a user's proximity, as their neighbors must have visited the same locations as the user in question in the past; however, some situations exist where pure collaborative filtering will yield meaningless recommendations. When a user travels to a different city, a collaborative filter will continue to output recommendations based on their previous location, potentially rendering the system useless [71] [77]. Even if a user were to not travel, it is possible to become neighbors with a travelling user, then begin to receive some recommendations for locations inaccessible for them. To account for these cases, we impose a weight on the ratings predicted by the collaborative filter based on the distance between candidate locations and the location that the user most recently visited. We calculate this weight with a Gaussian function, giving higher priority to locations close to the user:

$$\omega_{dist}\left(U_j, L_n\right) = \exp\left(-\frac{\sqrt{L_n^2 + L_{n-1}^2}}{2\left(\sigma\overline{d}\left(U_j\right)\right)^2}\right) \tag{4.6}$$

The numerator within the exponential function represents the distance between the candidate location L_n and the user's previously visited location L_{n-1}. $\overline{d}\left(U_j\right)$ is the mean value of the distance that user U_j travels between consecutive check-ins, and σ is a constant that controls the width of the Gaussian function determined empirically. This equation encapsulates the tendency of users to visit locations in the proximity of their previous check-ins [7] [71] [77] by decreasing the value assigned to more distant locations. As observed by Cho et al. [26], users exhibit both short and long-ranged movement patterns. To account for this, we calculate two values for $\overline{d}\left(U_j\right)$: the mean value of the distance travelled between all consecutively visited locations, and the mean value of the distance travelled between consecutively visited locations for which visits are within

a threshold time of each other. Then, when applying Eq. (4.6), the mean value between all consecutive visits is used when the time threshold between checkins is exceeded; else the value for check-ins within the threshold is employed. This is described by Eq. (4.7), where $T(L_n, L_{n-1})$ represents the time between check-ins at two locations, and τ represents the threshold time distinguishing between short and long-ranged movements.

$$\omega_{dist}(U_j, L_n) = \begin{cases} \exp\left(-\frac{\sqrt{L_n^2 + L_{n-1}^2}}{2\left(\sigma \overline{d_{short}}(U_j)\right)^2} \right) & T(L_n, L_{n-1}) < \tau \\ \exp\left(-\frac{\sqrt{L_n^2 + L_{n-1}^2}}{2\left(\sigma \overline{d_{all}}(U_j)\right)^2} \right) & T(L_n, L_{n-1}) \geq \tau \end{cases} \tag{4.7}$$

This weight is then used to scale predictions generated by Eq. (2.6):

$$pred_{dist}(U_j, L_n) = pred(U_j, L_n)\omega_{dist}(U_j, L_n) \tag{4.8}$$

The end result of integrating temporal, spatial, and social factors into a collaborative filter is a recommendation system governed by Eq. (2.2), Eq. (2.3), and Eq. (2.6), with modifications made to the rating value and similarity value according to Eq. (4.1) through Eq. (4.8). Combining all of these factors results in predictions generated according to Eq. (4.9):

$$pred_{context}(U_j, L_n) = \left(\overline{V}(U_j, L_{U_j}) + \frac{\sum_{x \in N_j} sim_{social}(U_j, x) \cdot \left(V(x,i) - \overline{V}(x, L_x) \right)}{\sum_{x \in N_j} sim_{social}(U_j, x)} \right) \omega_{dist}(U_j, L_n) \tag{4.9}$$

Our experiments focus on determining the effect that including each layer of context has on the quality of the recommendations generated both individually and as an ensemble.

4.3 Experimental Methodology

4.3.1 Datasets

We use four datasets and a subset of one of the datasets in our experiments. Two of the datasets and the subset were collected from several crawls of Foursquare performed by Gao et al. [25]. The other two datasets were collected from Brightkite and Gowalla by Cho et al [26]. Each dataset contains of a list of check-ins consisting of a user-location pair, a timestamp, and the latitude and longitude of the location. Also included is a social graph indicating which pairs of users are considered to be friends. Statistical information for each dataset is presented in Table 4.1.

Table 4.1: Dataset Statistics

Dataset	Start date	End date	Users	Locations	Check-ins	Social links
Social-Historical Ties data (SHTies) [25]	March 8, 2010	January 21, 2011	18 107	43 063	2 073 740	231 148
Geo-Social Correlation data (GScorr) [25]	January 1, 2011	July 31, 2011	11 326	182 968	1 385 223	47 164
Geo-Social Correlation subset [25]	January 1, 2011	March 31, 2011	5 269	26 381	288 079	10 208
Brightkite [26]	March 21, 2008	October 18, 2010	58 228	772 966	4 491 143	214 078
Gowalla [26]	February 24, 2009	October 23, 2010	196 591	1 280 969	6 442 890	950 327

4.3.2 Experimental setup and evaluation

The conditions for all of the experiments we perform are held constant. For each dataset, we take the first 80% of the data chronologically to serve as the training set, with the remaining 20% of the data designated as the test set. Each experiment is evaluated on how well predictions generated by the trained model align with the actual behaviour of each user on the test set. We select a variety of metrics to evaluate the performance of each experiment. For a test set of users

T, we calculate *precision* and *recall* of the top 10 recommendations generated by each model from a set of predicted locations for each user, P_{U_j}, and the set of locations visited by that user in the test set, L_{U_j}:

$$precision = \frac{1}{\|T\|} \sum_{U_j \in T} \frac{\left\| P_{U_j} \cap L_{U_j} \right\|}{\left\| P_{U_j} \right\|} \tag{4.10}$$

$$recall = \frac{1}{\|T\|} \sum_{U_j \in T} \frac{\left\| P_{U_j} \cap L_{U_j} \right\|}{\left\| L_{U_j} \right\|} \tag{4.11}$$

These measures give us a picture of how well our predictions match with the user's tastes (precision), and to what extent we are able to cover the set of relevant locations for each user (recall). While these values are useful in evaluating the quality of recommendations, there is no difference between the precision of a list which captures the most relevant locations in the top 3 spots and a list with irrelevant results at the top and 3 desired locations somewhere in the top 10. To evaluate this properly, we calculate the *mean average precision* [78] of the top-10 output. *Average precision* for each user is defined by Eq. (4.12), where *P(k)* is the precision for the top-*k* predictions:

$$AP(n) = \frac{\sum_{k=1}^{n} P(k)}{n} \tag{4.12}$$

Mean average precision, then, is simply the mean value of the average precision calculated for each user. This gives a heavier weight to predictions at the top of an ordered list.

Another metric we utilize is the rate at which the top prediction is correct for each time and location in the test set; this is also utilized by Gao et al. [7] [67]. While the other metrics generate

a single list of predictions and compare it with the actual locations visited in the test set, this method involves generating a prediction for each location in the test set. This approach allows us to evaluate the impact of incorporating spatial context into our predictions, as we generate a new prediction for each location in the test set.

For our experiments which consider the physical location of a user and the venues they visit, we use the *haversine formula* [79] to calculate the actual distance between venues from the longitude and latitude values provided in these datasets. This formula states that for any two points on a sphere, the haversine of their central angle is:

$$haversine\left(\frac{d}{r}\right) = haversine(lat_2 - lat_1) + \cos(lat_1)\cos(lat_2)\, haversine(long_2 - long_1)$$

(4.13)

where d is the distance between the points along the surface of the sphere, r is the radius of the sphere, lat_1, lat_2 indicate the latitude of locations 1 and 2, respectively, and $long_1$, $long_2$ indicate longitudes. The haversine function is calculated by:

$$haversine(\theta) = sin^2\left(\theta/_2\right) \tag{4.14}$$

This formula can then be rearranged to solve for the distance between points on the earth, d, as follows:

$$d = 2r \cdot arcsin\sqrt{sin^2\left(\frac{lat_2 - lat_1}{2}\right) + \cos(lat_1)\cos(lat_2)sin^2\left(\frac{long_2 - long_1}{2}\right)} \tag{4.15}$$

where r is the radius of the earth. It is important to note that this method of calculating distance is an approximation. The actual travel time between locations is affected by the topography of the surrounding area, and the means of travel available to the user. The meaning of 5 kilometers to someone travelling by car along a highway is considerably different compared to a person on

foot in a metropolitan area. Also, the distance calculated between locations suffers some inaccuracy due to the fact that the Earth is not a perfect sphere. Nonetheless, in the absence of any location data beyond latitude and longitude, this approach provides us a mechanism to model the distance between locations, though it is imperfect.

Prediction coverage [64] defines the degree to which a recommendation algorithm covers the inventory of items available for recommendation. Collaborative filtering algorithms are unable to make predictions for items that fall outside of the set rated by users in their neighborhood. An algorithm that is capable of highly accurate predictions on only a small fraction of the items available may be less desirable. For location-based recommendation, the prediction coverage of collaborative filtering is naturally quite poor, as a user is highly unlikely to form a neighborhood that would include physically distant locations. At the same time, recommendations for remote locations are meaningless if a user will never visit them. To capture a meaningful statistic for coverage in location-based social networks, we define it to be the fraction of locations for which our model can generate a ranking among the locations actually visited by each user. For a test set of users *T,* we calculate the coverage of a model using the set of locations that have been visited by a user's neighbors $L_{N_{U_j}}$ (that is, the set of locations that could be recommended to U_j) and the set of locations visited by that user in the test set, L_{U_j}:

$$\text{coverage} = \frac{\sum_{U_j \in T} \left\| L_{N_{U_j}} \cap L_{U_j} \right\|}{\sum_{U_j \in T} \left\| L_{U_j} \right\|} \tag{4.16}$$

4.4 Experimental Results

This section provides the results of our experiments, showing the improvements in recommendation quality as a result of including context in a collaborative filter. We compare the results of incorporating contextual data into our model against a baseline collaborative filter and against other works that utilize context to perform recommendations on the same datasets. We first examine the performance of a user-user collaborative filter that does not incorporate any contextual data in Table 4.2.

Table 4.2: User-user collaborative filter

Dataset	Neighborhood size	MAP	Precision	Recall	Top-1	Coverage
SHTies	20	0.3880	**0.1582**	**0.2965**	0.1120	0.6988
	30	**0.3883**	0.1570	0.2930	**0.1138**	0.7313
	50	0.3843	0.1541	0.2856	0.1105	0.7657
	100	0.3728	0.1485	0.2718	0.1051	0.8011
	200	0.3636	0.1431	0.2601	0.1015	0.8323
	300	0.3570	0.1397	0.2533	0.0986	**0.8478**
GSCorr	20	**0.6180**	**0.1785**	**0.2583**	**0.2012**	0.6632
	30	0.6150	0.1774	0.2580	0.1966	0.6926
	50	0.6094	0.1747	0.2556	0.1926	0.7266
	100	0.5971	0.1685	0.2483	0.1839	0.7679
	200	0.5827	0.1622	0.2405	0.1762	0.8020
	300	0.5748	0.1589	0.2367	0.1723	**0.8194**
GSCorr subset	20	**0.4674**	0.2013	0.0903	**0.0685**	0.5215
	30	0.4651	**0.2028**	0.0913	0.0666	0.5553
	50	0.4591	0.2023	**0.0917**	0.0642	0.5962
	100	0.4541	0.1968	0.0900	0.0634	0.6423
	200	0.4488	0.1911	0.0880	0.0621	0.6801
	300	0.4448	0.1878	0.0871	0.0615	**0.6998**
Brightkite	20	0.2009	0.0315	0.1406	0.1549	0.4199
	30	0.2032	0.0320	0.1423	0.1551	0.4326
	50	0.2045	0.0323	0.1433	0.1554	0.4470
	100	0.2062	0.0327	0.1443	**0.1556**	0.4648
	200	0.2086	0.0329	0.1445	0.1555	0.4814

	300	**0.2096**	**0.0329**	**0.1453**	0.1554	**0.4984**
Gowalla	20	0.1325	0.0354	0.0444	0.0150	0.2542
	30	**0.1332**	0.0367	0.0459	**0.0150**	0.2820
	50	0.1330	0.0378	0.0472	0.0147	0.3156
	100	0.1322	**0.0382**	**0.0477**	0.0144	0.3582
	200	0.1313	0.0377	0.0474	0.0139	**0.3939**
	300	0.1310	0.0374	0.0473	0.0136	**0.4101**

Across all datasets, prediction coverage steadily increases alongside neighborhood size. This result is expected, as increasing the size of a neighborhood can only add to the set of items that can be ranked. Among the Foursquare datasets, SHTies and GSCorr have similar scores for coverage, but there is a noticeable drop off for the GSCorr subset. This exemplifies the nature of the cold-start problem [21]: even though the data is the same as the data in GSCorr, simply the fact that there are fewer entries causes a marked decrease in coverage. This effect extends to most of the other measures used, though the precision on the GSCorr subset is actually superior to that found on the full dataset. A possible cause of this is that in the full GSCorr dataset, there are more locations relative to the number of users with 16.15 locations per user, while the subset has only 5.00 locations per user; thus, we see higher precision on the subset because there are fewer incorrect locations to predict. The fact that the mean average precision remains higher for the full set of data indicates that, while the subset has a higher ratio of correct predictions in the top 10, the overall rankings are more accurate for predictions based on the full dataset. Additionally, as the neighborhood size increases, all three Foursquare datasets begin to show diminishing returns of MAP, precision, recall, and the accuracy of the top prediction. These results exemplify a trade-off inherent in parameterizing a collaborative filter -- up to a certain point, adding neighbors is beneficial, but eventually, distant neighbors will begin to lead predictions astray. Comparing the results on Gowalla against the other datasets, we notice a

significant drop-off in accuracy -- particularly in top-1 prediction accuracy. This can be explained by examining the amount of data present for each user: from the statistics in Table 4.1, we can see that the mean amount of check-ins for each Gowalla user is 33, while Brightkite has a mean value of 77 check-ins per user. The reduction in accuracy indicates that the Gowalla dataset suffers from the cold-start problem [21].

In Table 4.3, we add the first layer of context to our recommendation algorithm, incorporating weights based on the time and frequency of visits as described by Eq. (4.2).

Table 4.3: Temporal Context

Dataset	Neighborhood size	MAP	Precision	Recall	Top-1	Coverage
SHTies	20	**0.4480**	0.1507	0.2882	**0.1892**	0.6941
	30	0.4426	**0.1514**	**0.2889**	0.1846	0.7277
	50	0.4351	0.1512	0.2866	0.1809	0.7647
	100	0.4264	0.1493	0.2807	0.1720	0.8050
	200	0.4179	0.1467	0.2745	0.1616	0.8376
	300	0.4137	0.1453	0.2714	0.1564	**0.8524**
GSCorr	20	**0.6624**	0.1584	0.2407	**0.2479**	0.6312
	30	0.6576	0.1605	0.2435	0.2436	0.6656
	50	0.6547	**0.1607**	**0.2441**	0.2402	0.7080
	100	0.6502	0.1600	0.2430	0.2333	0.7580
	200	0.6458	0.1590	0.2416	0.2254	0.7970
	300	0.6435	0.1582	0.2403	0.2123	**0.8163**
GSCorr subset	20	**0.5080**	0.1817	0.0831	**0.0951**	0.5193
	30	0.5058	0.1855	0.0843	0.0938	0.5546
	50	0.5010	**0.1893**	**0.0857**	0.0923	0.5972
	100	0.4970	0.1892	0.0856	0.0905	0.6458
	200	0.4927	0.1873	0.0850	0.0886	0.6839
	300	0.4902	0.1860	0.0845	0.0854	**0.7019**
Brightkite	20	0.2247	0.0304	0.1423	0.1748	0.4270
	30	0.2249	0.0307	0.1430	0.1747	0.4369
	50	**0.2251**	0.0311	0.1439	**0.1770**	0.4527
	100	0.2242	0.0313	**0.1445**	0.1761	0.4748

	200	0.2229	**0.0318**	0.1454	0.1757	0.5027
	300	0.2228	0.0317	0.1454	0.1750	**0.5103**
Gowalla	20	**0.1504**	0.0358	0.0454	**0.0244**	0.2693
	30	0.1503	0.0370	0.0465	0.0237	0.2953
	50	0.1493	0.0381	0.0478	0.0229	0.3278
	100	0.1475	0.0388	0.0484	0.0215	0.3674
	200	0.1460	**0.0388**	**0.0485**	0.0211	0.3999
	300	0.1455	0.3864	0.4844	0.0209	**0.4147**

We observe similar effects to Table 4.2 as neighborhood size changes in Table 4.3. Increasing the neighborhood size will always improve the range of predictions that can be generated, and up to a point, the quality of the predictions also improves. Precision and recall in particular tend to peak at larger neighborhood sizes than MAP and top-1 accuracy, indicating that while a larger neighborhood becomes better at getting relevant results into a top-10 list, smaller neighborhoods tend to be better at ordering that list.

Table 4.4 presents the results of integrating social context into our recommendation algorithm as defined by Eq. (4.5). We investigate a range of values for α to find a balance between placing greater weight on friends and not excluding relevant non-friended users. The neighborhood size for every experiment in Table 4.4 is held constant at 30.

Table 4.4: Social Context

Dataset	α	MAP	Precision	Recall	Top-1	Coverage
SHTies	1.00	0.3887	0.1570	0.2929	0.1532	0.7312
	0.75	0.3901	0.1574	0.2938	0.1538	0.7338
	0.50	0.3914	**0.1579**	**0.2949**	0.1567	0.7372
	0.25	0.3946	0.1575	0.2943	0.1572	0.7378
	0.10	0.3973	0.1564	0.2924	0.1612	0.7393
	0.05	**0.3974**	0.1541	0.2881	0.1613	0.7403
	0.01	0.3949	0.1472	0.2760	**0.1625**	**0.7420**
	0.00	0.2000	0.0737	0.1365	0.0753	0.3352

GSCorr	1.00	0.6148	0.1774	0.2582	0.2478	0.6925
	0.75	0.6162	0.1777	0.2587	0.2483	0.6936
	0.50	0.6186	**0.1778**	**0.2591**	0.2489	0.6949
	0.25	0.6227	0.1778	0.2591	0.2510	**0.6959**
	0.10	0.6290	0.1762	0.2581	0.2527	0.6963
	0.05	0.6336	0.1741	0.2559	0.2533	0.6962
	0.01	**0.6386**	0.1674	0.2484	**0.2545**	0.6961
	0.00	0.3109	0.0701	0.1085	0.1188	0.3267
GSCorr subset	1.00	0.4649	0.2029	0.0913	0.1126	0.5557
	0.75	0.4654	0.2034	**0.0915**	0.1126	0.5564
	0.50	0.4659	0.2032	0.0915	0.1130	0.5568
	0.25	0.4673	**0.2033**	0.0913	0.1136	0.5576
	0.10	0.4692	0.2029	0.0913	0.1144	0.5591
	0.05	0.4697	0.2013	0.0907	0.1145	0.5598
	0.01	**0.4708**	0.1935	0.0878	**0.1147**	**0.5608**
	0.00	0.1497	0.0531	0.0236	0.0334	0.1544
Brightkite	1.00	0.2029	0.0319	0.1422	0.2756	0.4323
	0.75	0.2033	0.0322	0.1433	0.2744	0.4347
	0.50	0.2050	**0.0323**	0.1433	0.2746	0.4359
	0.25	0.2054	0.0322	**0.1437**	0.2757	0.4362
	0.10	0.2068	0.0320	0.1430	0.2766	0.4417
	0.05	0.2088	0.0318	0.1427	**0.2760**	0.4414
	0.01	**0.2113**	0.0313	0.1407	0.2753	**0.4443**
	0.00	0.1063	0.0169	0.0489	0.1161	0.2246
Gowalla	1.00	0.1332	0.0367	0.0459	0.0288	0.2819
	0.75	0.1333	0.0368	0.0461	0.0290	0.2822
	0.50	0.1337	0.0370	0.0463	0.0294	0.2830
	0.25	0.1346	0.0372	0.0465	0.0301	0.2846
	0.10	0.1360	**0.0373**	**0.0468**	0.0314	0.2867
	0.05	0.1369	0.0372	0.0466	0.0328	0.2879
	0.01	**0.1378**	0.0360	0.0455	**0.0348**	**0.2901**
	0.00	0.0490	0.0140	0.0152	0.0174	0.1032

Examining the effect of varying the social weight coefficient, we find that going to the extremes has a negative effect on the performance of the model. Giving no additional weight to friends ($\alpha = 1.00$) does not perform as well as values in the range [0.01, 0.75]. Setting the coefficient to

zero, Ignoring recommendations from similar users who are not friends ($\alpha = 0.00$) has a drastic impact on the quality of the results, yielding far worse performance than the unmodified collaborative filter. It is of interest to note that coverage increases as the social weight coefficient approaches (but does not reach) 0. In traditional collaborative filtering, a user's neighborhood consists of other users who have similar location histories. By placing a heavy weight on users who have been explicitly specified as friends, a user's neighborhood may contain a friend whose location history does not heavily overlap. This case would be unlikely to be present in unmodified collaborative filtering, as dissimilar users would be unlikely to become neighbors. However, a heavy social weight would increase the chance of this happening, leading to the results we see here. Overall, social links between users can be used to improve recommendation quality, but the groundwork laid by collaborative filtering should not be discarded; the similarity between users who are not explicitly friends is still very useful in generating recommendations.

Physical location is the final layer of context we examine. The focus of our experiments here is to find a value for the parameter σ from Eq. (4.7). The value of τ is held constant at 12 hours for these experiments. We evaluate this approach by generating the top recommendation for each location in the test set sequentially, allowing us to use the distance from the previous location to weight our predictions according to Eq. (4.8). The results of this method are summarized in Table 4.5.

Table 4.5: Spatial Context (top-1 accuracy)

Dataset	σ	Neighborhood size		
		30	50	100
SHTies	0.001	0.3867	0.3998	0.4101
	0.01	**0.3884**	**0.4013**	**0.4113**
	0.1	0.2566	0.2587	0.2558
	1	0.1607	0.1601	0.1570

	10	0.1571	0.1569	0.1543
GSCorr	0.001	0.2919	0.2984	0.3061
	0.01	0.2946	0.3009	0.3081
	0.1	**0.3066**	**0.3101**	**0.3137**
	1	0.2535	0.2553	0.2561
	10	0.2518	0.2539	0.2548
GSCorr Subset	**0.001**	**0.2381**	**0.2473**	**0.2555**
	0.01	0.2364	0.2442	0.2518
	0.1	0.1629	0.1643	0.1656
	1	0.1175	0.1178	0.1196
	10	0.1170	0.1169	0.1191
Brightkite	0.001	0.3364	0.3441	0.3529
	0.01	**0.3458**	**0.3516**	**0.3626**
	0.1	0.3254	0.3330	0.3374
	1	0.3016	0.3020	0.3070
	10	0.3014	0.3035	0.3071
Gowalla	0.001	0.0374	0.0384	0.0398
	0.01	**0.0381**	**0.0390**	**0.0399**
	0.1	0.0345	0.0343	0.0346
	1	0.0317	0.0316	0.0316
	10	0.0316	0.0316	0.0315

Across all datasets except for the GSCorr subset, we find that the best accuracy is to be had with the value of σ set to 0.01. A coefficient of 1 would involve weighting locations around a user according to a Gaussian function with the standard deviation equal to the mean travel distance of the user; it is interesting to note that we find better performance at a much smaller value. The practical meaning of this is that when a user visits locations sequentially, it is likely that that they will visit several in the same small area before travelling a greater distance to a new set of locations (increasing the mean distance between locations). Thus, we can improve the quality of our model by focusing our predictions on locations in the immediate vicinity of a user.

Table 4.6 presents a comparison of our results from Tables 2-5 with a combination of all contexts and the work done by Gao et al. [7], [68], [67] and Wang et al. [71]. Each of these previous articles only considers a single dataset.

Table 4.6: Comparison

Dataset	Model	MAP	Precision	Recall	Top-1	Coverage
SHTies	Basic CF	0.3883	0.1570	0.2930	0.1138	0.7313
	Temporal	**0.4426**	0.1514	0.2889	0.1846	0.7277
	Social	0.3946	**0.1575**	**0.2943**	0.1572	**0.7378**
	Spatial	-	-	-	0.3884	0.7313
	Combination	0.4424	0.1526	0.2906	**0.3936**	0.7333
	Social-historical [68]	-	-	-	0.3423	-
GSCorr	Basic CF	0.6150	0.1774	0.2580	0.1966	0.6926
	Temporal	0.6576	0.1605	0.2435	0.2436	0.6656
	Social	0.6227	**0.1778**	**0.2591**	0.2510	**0.6959**
	Spatial	-	-	-	**0.3066**	0.6926
	Combination	**0.6614**	0.1615	0.2451	0.2925	0.6718
	Geo-social correlation [7]	-	-	-	0.1921	-
GSCorr Subset	Basic CF	0.4651	0.2028	0.0913	0.0666	0.5553
	Temporal	0.5058	0.1855	0.0843	0.0938	0.5546
	Social	0.4673	**0.2033**	**0.0913**	0.1136	0.5576
	Spatial	-	-	-	**0.2381**	0.5553
	Combination	**0.5066**	0.1862	0.0844	0.2370	**0.5589**
	LRT [67]	-	0.0300	0.0335	-	-
Brightkite	Basic CF	0.2032	0.0320	0.1423	0.1551	0.4326
	Temporal	0.2249	0.0307	0.1430	0.1747	0.4369
	Social	0.2054	**0.0322**	**0.1437**	0.2757	0.4362
	Spatial	-	-	-	0.3458	0.4326
	Combination	**0.2253**	0.0314	0.1426	**0.3601**	**0.4511**
	LFBCA [71]	-	0.0220	0.0300	-	-
Gowalla	Basic CF	0.1332	0.0367	0.0459	0.0150	0.2820
	Temporal	0.1503	0.0370	0.0465	0.0237	0.2953
	Social	0.1346	0.0372	0.0465	0.0301	0.2846
	Spatial	-	-	-	0.0381	0.2820
	Combination	**0.1517**	**0.0380**	**0.0482**	**0.0426**	**0.3330**
	LFBCA [71]	-	0.0370	0.0430	-	-

Comparing basic collaborative filtering to the collaborative filter incorporating temporal context, the main improvements are visible in mean average precision and top-1 accuracy. Precision and recall tend to suffer slightly, and prediction coverage remains largely unchanged. A decrease in precision alongside an increase in mean average precision indicates that while users visited less of the predicted top-10 locations, predictions near the top of the list are correct more often with temporal context than without. Incorporating social context offers improvements over the unmodified collaborative filter across all evaluation metrics. Social context outperforms temporal context in terms of precision and recall across all datasets, while temporal context results in better MAP. Incorporating spatial context into the collaborative filter results in a drastic increase in top-1 accuracy over all other models examined. Filtering out distant locations and focusing predictions on locations physically close to the user improves the quality of recommendations considerably. The combined model integrates all three layers of context, and produces superior values of mean average precision and top-1 accuracy across most datasets (though the individual temporal/spatial approaches outperform the combination in some cases). The reduction in precision and recall from the temporal filter is still present in the combined model; it is mitigated somewhat by the presence of the social filter, but the social-only approach still yields higher scores for precision and recall across most datasets (with the exception of Brightkite). Comparing the results of the combined model against previously developed methods evaluated on these five datasets, we find that our method offers superior performance across all comparable evaluation metrics.

4.5 Summary

In this chapter, we have studied the inclusion of contextual factors in recommendation algorithms for location-based social networks. We propose a framework to improve predictive

accuracy by augmenting a user-user collaborative filter with spatial, temporal, and social contextual data. Spatial data is used to restrict recommendations to locations that a user could feasibly access. Incorporating temporal information allows us to account for the change in a user's preferences over time, discounting locations they may have visited in the past that no longer fall within their interests. Social data is applied to increase the weight on locations that a user's friends have visited, as it is possible they would enjoy exploring the interests of each other, or attending a common location as a group. We have examined the effects that each of these layers of context have on the quality of recommendations produced, both individually and in combination with each other. The effect of parameters related to each layer of context were explored, and the best parameterizations were merged to create a final model. The combined effect of incorporating all three layers of context improves the performance of a basic collaborative filter to be more accurate than algorithms previously designed for the five datasets tested.

5 Summary and Future Work

In this thesis, we have investigated recommendation algorithms for integrated library systems and location-based social networks. Collaborative filtering algorithms struggle in the face of sparse data; we strive to overcome the cold-start problem by incorporating content and context data into the recommendation process. For integrated library systems, we developed an ensemble classifier to merge the results of an item-item collaborative filter with a content filter employing "fuzzy taste vectors". Experiments on two benchmark datasets indicate that this approach is as accurate as any other fuzzy recommender on denser datasets, and superior on sparser datasets. Thus, we conclude that the fuzzy taste vector algorithm appears to be effective in overcoming the cold-start problem; however, it is dependent on having access to high-quality, detailed content data. For location-based social networks, we proposed methods to incorporate temporal, social, and spatial contextual factors into a user-based collaborative filter. We evaluated our context-aware recommender on five benchmark datasets, and found that our approach offers superior predicative accuracy to other recommendation algorithms evaluated on these datasets. In general, we find that including content and context information to supplement the user-item matrix allows us to improve the quality of recommendations generated, though the ability to do so is subject to having access to sufficient additional data.

Future work in this area includes a more thorough analysis of potential methods for blending content and contextual data into a recommendation engine. More functions for generating weights associated with spatial, temporal, and social factors could be the subject of experimentation, such as taking into account the time of day or week when considering time, or

considering the effect that groups of friends have on each other collectively, rather than in a one-to-one configuration. Our experiments could be expanded to include more datasets, and more content and contextual data could be included. In particular, a combination of data concerning users and items/locations could be merged into the same model, as current experiments focus only on user-based context data or item-based content data. Studying the effect that recommendations provided by the system have on user behaviour in a live experiment would also be of great interest. Offline datasets are limited to predicting how a user will behave in the absence of a recommender system; analyzing a system in active use would allow us to evaluate recommendations in terms of how often users follow them. For location-based social networks, live experiments would also open the door to use the current location of a user as an input to the recommendation system, rather than simulating their location based on their past check-ins.

6 References

[1] eMarketer, "Ecommerce Sales Topped $1 Trillion for First Time in 2012," eMarketer, 2013. [Online]. Available: http://www.emarketer.com/Article/Ecommerce-Sales-Topped-1-Trillion-First-Time-2012/1009649. [Accessed 15 July 2013].

[2] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Analysis of recommendation algorithms for e-commerce," *Proceedings of the 2nd ACM conference on Electronic commerce,* 2000.

[3] P. Chen and S. Wu, "Does collaborative filtering technology impact sales? Empirical evidence from Amazon. com," *Empirical Evidence from Amazon. Com (July 8, 2007),* 2007.

[4] O. Celma and P. Lamere, "Music recommendation tutorial," *ISMIR. Vienna, Austria,* 2007.

[5] K. Zickuhr, "Three-quarters of smartphone owners use location-based services," Pew Internet & American Life Project, 2012. [Online]. Available: http://www.pewinternet.org/2012/05/11/three-quarters-of-smartphone-owners-use-location-based-services/. [Accessed 2 June 2014].

[6] Y. Zheng, Y. Chen, X. Xie and W. Ma, "GeoLife2. 0: a location-based social networking service," *MDM '09 Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware,* 2009.

[7] H. Gao, J. Tang and H. Liu, "gSCorr: Modeling geo-social correlations for new check-ins on location-based social networks," *Proceedings of the 21st ACM international conference on Information and knowledge management,* pp. 2-6, 2012.

[8] Foursquare, "About Foursquare," Foursquare, 2014. [Online]. Available: https://foursquare.com/about. [Accessed 20 April 2014].

[9] M. Ye, P. Yin and W. Lee, "Location recommendation for location-based social networks," *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems,* pp. 458-461, 2010.

[10] M. Ye, P. Yin, W. Lee and D. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11,* p. 325, 2011.

[11] C. Sisco and A. Stonebridge, "Toward Thriving Northern Communities," *Ottawa, ON, Canada: Conference Board of Canada,* 2010.

[12] T. Claridge, "Definitions of Social Captial," 2004.

[13] J. Freyne, M. Jacovi, I. Guy and W. Geyer, "Increasing engagement through early recommender

intervention," *Proceedings of the third ACM conference on Recommender systems,* pp. 85-92, 2009.

[14] S. Abbar, M. Bouzeghoub and S. Lopez, "Context-Aware Recommender Systems : A Service-Oriented Approach," *VLDB PersDB Workshop,* pp. 1-6, 2009.

[15] J. Schafer, D. Frankowski, J. Herlocker and S. Sen, "Collaborative filtering recommender systems," *The adaptive web,* pp. 291-324, 2007.

[16] D. Jannach, M. Zanker, A. Felfernig and G. Friedrich, Recommender systems: an introduction, Cambridge University Press, 2010.

[17] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Item-based collaborative filtering recommendation algorithms," *Proceedings of the 10th international conference on World Wide Web,* 2001.

[18] P. Lops, M. De Gemmis and G. Semeraro, Recommender Systems Handbook, Boston, MA: Springer US, 2011.

[19] M. Pazzani and D. Billsus, "Content-based recommendation systems," *The adaptive web,* pp. 325-341, 2007.

[20] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender systems handbook*, Springer US, 2011, pp. 217-253.

[21] A. Schein, A. Popescul, L. Ungar and D. Pennock, "Methods and metrics for cold-start recommendations," *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval,* pp. 253-260, 2002.

[22] Y. Cao and Y. Li, "An intelligent fuzzy-based recommendation system for consumer electronic products," *Expert Systems with Applications,* vol. 33, no. 1, pp. 230-240, 2007.

[23] C. Ziegler, S. McNee, J. Konstan and G. Lausen, "Improving recommendation lists through topic diversification," *Proceedings of the 14th international conference on World Wide Web - WWW '05,* p. 22, 2005.

[24] "MovieLens Data Sets," 2011. [Online]. Available: http://www.grouplens.org/node/73. [Accessed 18 March 2013].

[25] H. Gao and H. Liu, "Location-Based Social Network Data Repository," 2014. [Online]. Available: http://www.public.asu.edu/~hgao16/dataset.html. [Accessed 1 May 2014].

[26] E. Cho, S. Myers and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining,* 2012.

[27] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of

the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering,* vol. 17, no. 6, pp. 734-749, 2005.

[28] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," *Recommender Systems Handbook,* 2011.

[29] C. Piao, J. Zhao and L. Zheng, "Research on entropy-based collaborative filtering algorithm and personalized recommendation in e-commerce," *Service Oriented Computing and Applications,* vol. 3, no. 2, pp. 147-157, 2008.

[30] P. Jaccard, "Etude comparative de la distribution florale dans une portion des Alpes et du Jura," *Impr. Corbaz,* 1901.

[31] A. Shepitsen, J. Gemmell, B. Mobahser and R. Burke, "Personalized recommendation in social tagging systems using hierarchical clustering," *Proceedings of the 2008 ACM conference on Recommender systems,* pp. 259-266, 2008.

[32] T. Zhou, L. Jiang, R. Su and Y. Zhang, "Effect of initial configuration on network-based recommendation," *EPL (Europhysics Letters),* vol. 81, no. 5, 2008.

[33] Z. Zaier, R. Godin and L. Faucher, "Evaluating Recommender Systems," *2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution,* pp. 211-217, 2008.

[34] D. Agarwal and B. Chen, "fLDA: matrix factorization through latent dirichlet allocation," *Proceedings of the third ACM international conference on Web search and data mining,* 2010.

[35] A. Sieg, B. Mobasher and R. Burke, "Improving the effectiveness of collaborative recommendation with ontology-based user profiles," *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems - HetRec '10,* pp. 39-46, 2010.

[36] G. Abowd, A. Dey and P. Brown, "Towards a better understanding of context and context-awareness," *Handheld and ubiquitous computing,* 1999.

[37] J. Herlocker and J. Konstan, "Content-independent task-focused recommendation," *IEEE Internet Computing,* vol. 5, no. 6, pp. 40-47, 2001.

[38] G. Adomavicius and A. Tuzhilin, "Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach," *ACM Transactions on Information Systems,* vol. 23, no. 1, pp. 103-145, 2005.

[39] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," *Proceedings of ACM SIGIR Workshop on Recommender System,* 1999.

[40] S. Shockaert, N. Makarytska and M. Cock, "Fuzzy methods on the web: a critical discussion," *35 Years of Fuzzy Set Theory,* pp. 237-266, 2011.

[41] H. Hino, Y. Fujimoto and N. Murata, "A grouped ranking model for item preference parameter," *Neural computation,* pp. 2417-2451, 2010.

[42] H. Sahbi and N. Boujemaa, "Fuzzy Clustering : Consistency of Entropy Regularization ∗ 1 Introduction," *Computational Intelligence, Theory and Applications,* pp. 95-107, 2004.

[43] K. Treerattanapitak and C. Jaruskulchai, "Exponential Fuzzy C-Means for Collaborative Filtering," *Journal of Computer Science and Technology,* vol. 27, no. 3, pp. 567-576, 2012.

[44] R. Forsati, H. Mohammadi, M. Shamsfard and A. Keikha, "A fuzzy co-clustering approach for hybrid recommender systems," *International Journal of Hybrid Intelligent Systems,* vol. 10, no. 2, pp. 71-81, 2013.

[45] M. Krishna and K. Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 29, no. 3, pp. 433-439, 1999.

[46] H. Guo, X. Zhu, K. Gao and T. Liu, "An improved genetic k-means algorithm for optimal clustering," *Sixth IEEE International Conference on Data Mining-Workshops,* 2006.

[47] J. Bobadilla, F. Ortega, A. Hernando and J. Alcalá, "Improving collaborative filtering recommender system results and performance using genetic algorithms," *Knowledge-Based Systems,* vol. 24, no. 8, pp. 1310-1316, 2011.

[48] R. Capuruço and L. Capretz, "A fuzzy-based inference mechanism of trust for improved social recommenders," *UMAP Workshops,* 2012.

[49] D. Wu, G. Zhang and J. Lu, "A Fuzzy Tree Similarity Based Recommendation Approach for Telecom Products," *Proceedings of the 2013 Joint IFSA World Congress NAFIPS Annual Meeting,* pp. 813-818, 2013.

[50] V. Bhavsar, H. Boley and L. Yang, "A Weighted-Tree Similarity Algorithm for Multi-Agent Systems in E- Business Environments," *Computational Intelligence,* vol. 20, 2004.

[51] C. Cornelis, X. Guo, J. Lu and G. Zhang, "A Fuzzy Relational Approach to Event Recommendation," *Indian International Conference on Artificial Intelligence,* pp. 2231-2242, 2005.

[52] C. Cornelis, J. Lu, X. Guo and G. Zhang, "One-and-only item recommendation with fuzzy logic techniques," *Information Sciences,* vol. 177, no. 22, pp. 4906-4921, 2007.

[53] C. Porcel and A. Tejeda-Lorente, "A hybrid recommender system for the selective dissemination of research resources in a Technology Transfer Office," *Information Sciences,* vol. 184, no. 1, pp. 1-19, 2012.

[54] C. Porcel, A. López-Herrera and E. Herrera-Viedma, "A recommender system for research resources based on fuzzy linguistic modeling," *Expert Systems with Applications,* vol. 36, no. 3, pp. 5173-5183, 2009.

[55] C. Porcel and M. Cobo, "An improved recommender system to avoid the persistent information overload in a university digital library," *Control and Cybernetics,* vol. 39, no. 4, 2010.

[56] J. Morales-del-castillo, E. Pes and E. Herrera-Videma, "A filtering and recommender system for e-scholars," *nternational Journal of Technology Enhanced Learning,* vol. 2, no. 3, pp. 227-240, 2010.

[57] N. Matsatsinis, K. Lakiotaki and P. Delia, "A system based on multiple criteria analysis for scientific paper recommendation," *Proceedings of the 11th Panhellenic Conference on Informatics,* 2007.

[58] Z. Haung, W. Chung, T. Ong and H. Chen, "A graph-based recommender system for digital library," *Proceedings of the second ACM/IEEE-CS joint conference on Digital libraries - JCDL '02,* p. 65, 2002.

[59] J. Herlocker, S. Jung and J. Webster, "Collaborative filtering for digital libraries," 2012.

[60] A. Zenebe, L. Zhou and A. Norcio, "User preferences discovery using fuzzy models," *Fuzzy Sets and Systems,* vol. 161, no. 23, pp. 3044-3063, 2010.

[61] "Amazon Web Services," 2012. [Online]. Available: http://aws.amazon.com/. [Accessed 22 Nov 2012].

[62] "Library of Congress," 2012. [Online]. Available: http://www.loc.gov. [Accessed 15 October 2012].

[63] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *International Joint Conference on Artificial Intelligence,* 1995.

[64] J. Herlocker, J. Konstan, L. Terveen and J. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems,* vol. 22, no. 1, pp. 5-53, 2004.

[65] H. Kwon, T. Lee and K. Hong, "Improved Memory-based Collaborative Filtering Using Entropy-based Similarity Measures," *Proceedings of the 2009 International Symposium on Web Information Systems and Applications,* pp. 29-34, 2009.

[66] J. Bao, Y. Zheng, D. Wilkie and M. F. Mokbel, "A Survey on Recommendations in Location-based Social Networks," *Submitted to GeoInformatica,* 2014.

[67] H. Gao, J. Tang, X. Hu and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks," *Proceedings of the 7th ACM conference on Recommender systems,* pp. 93-100, 2013.

[68] H. Gao, J. Tang, X. Hu and H. Liu, "Modeling temporal effects of human mobile behavior on

location-based social networks," *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management,* pp. 1673-1678, 2013.

[69] L. Song, D. Kotz, R. Jain and X. He, "Evaluating location predictors with extensive Wi-Fi mobility data," *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies,* vol. 2, pp. 1414-1424, 2004.

[70] H. Gao, J. Tang and H. Liu, "Exploring Social-Historical Ties on Location-Based Social Networks," *ICWSM,* 2012.

[71] H. Wang and N. Mamoulis, "Location Recommendation in Location-based Social Networks using User Check-in Data," *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* 2013.

[72] G. Jeh and J. Widom, "Scaling personalized web search," *Proceedings of the 12th international conference on World Wide Web,* pp. 271-279, 2003.

[73] M. Ye, W. Lee and C. Mulligann, "What you are is When you are : The Temporal Dimension of Feature Types in Location-based Social Networks," *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* pp. 102-111, 2011.

[74] J. Bao, Y. Zheng and M. Mokbel, "Location-based and preference-aware recommendation using sparse geo-social networking data," *Proceedings of the 20th International Conference on Advances in Geographic Information Systems,* p. 199, 2012.

[75] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM,* vol. 53, no. 4, p. 89, 2010.

[76] Y. Ding and X. Li, "Time weight collaborative filtering," *Proceedings of the 14th ACM international conference on Information and knowledge management,* 2005.

[77] G. Ference, M. Ye and W. Lee, "Location recommendation for out-of-town users in location-based social networks," *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management,* pp. 721-726, 2013.

[78] Kaggle, "Mean Average Precision," Kaggle, 2014. [Online]. Available: https://www.kaggle.com/wiki/MeanAveragePrecision. [Accessed 20 May 2014].

[79] C. C. Robusto, "The cosine-haversine formula," *American Mathematical Monthly,* pp. 38-40, 1957.