UNIVERSITY OF SOUTHAMPTON

# How to recommend music to film buffs:

**Enabling the provision of recommendations from multiple domains.**

by

Antonis Loizou

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

May 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Antonis Loizou

In broad terms, Recommender Systems use machine learning techniques to process historical data about their user's interests, encoded in user profiles. Once the algorithms used have been trained on user profiles, their output is used to compile a ranked list of all resources available for recommendation, based on each profile.

Collaborative Filtering is the most widespread method of carrying this out, building on the intuition that similar people will be interested in the same things. The point of failure in this approach lies in that similarity can only be assessed between users that have expressed their preferences on a common set of resources. This requirement prohibits the sharing of preference data across different systems, and causes additional problems when new resources for recommendation become available, or when new users subscribe to the system.

I propose that the difficulty can be overcome by identifying and exploiting semantic relationships between the resources available for recommendation themselves. Moreover, systems that are able to assess the strength of the relationship between any two resources can provide recommendations from multiple domains. For example, music recommendations can be made based on a person's film taste if strong semantic relationships can be identified between certain films and the music he/she listens to.

As such the contributions made by this dissertation can be summarised in the following:

1. **Facilitating the comparison of heterogeneous resources**

   The use of Wikipedia is proposed for this purpose, under the assumption that hyper–links between articles in Wikipedia convey latent semantic relationships between the concepts they describe. Thus, a methodology for projecting domain resources onto Wikipedia has been developed. The assumption is then validated by showing evidence that the projections are successful in retaining similarity between domain resources, in three independent domains.

2. **Enabling the provision of recommendations from multiple domains**

   The aforementioned projections encode the links present in Wikipedia articles that are found to correspond to domain resources, and can be viewed collectively as a graph. In addition, the Internet is populated with social networks of people who express their preferences on a given set of resources in the form of ratings. Members of such communities are included as nodes in the graph and ratings regarding domain resources represented as edges. A reversible Markov chain model was implemented to describe the probabilities associated with the traversal of edges in the integrated graph. Nodes that represent resources and other concepts the user is known to be interested in are then identified in the graph. Using these nodes as a starting point, the resource nodes most likely to be reached after an arbitrarily large number of edge traversals are considered the most relevant to the user and are recommended. Experimental results show that the framework is successful in predicting user preferences in domains different to those of the input.

# Contents

# List of Figures

# List of Tables

# Acknowledgements

I would like to express my gratitude to my primary supervisor, Dr. Srinandan Dasmahapatra, for his invaluable support and guidance in pursuing my PhD. My research has benefited greatly from his constructive criticism, and his ability to point me in the right direction each time I felt I had reached a dead end. I would also like to thank my second superivisor, Prof. Paul H. Lewis, whose expertise, understanding, and patience, added considerably to my graduate experience. In short, I could not have asked for a better supervisory team, and I feel priviliged to have worked with them.

Additionally, I am grateful for the help of a number of colleagues and friends. I thank Martin Szomszor and Harith Alani for providing me with the necessary data to evaluate the performance of my framework. Moreover, Harith's input as my undergraduate project supervisor was one of the main motivating factors for me to apply for a PhD in the first place. I also acknowledge the contrubutions of Mischa Tuffield, with whom I collaborated in developing the Semantic Logger system, which was later used as the user profiling component of the framework. Also, I greatly appreciate Sebastien Francois' help in patiently explaining the basics of Perl, and providing me with various snippets of code. Lastly, I'd like to thank Simone Scaringi for his constant scepticism regarding my experimental results. Beeing in a state of brotherly competition with someone that passionate about science can really push one's limits.

My work has been supported financially by the OpenKnowledge EU Framework 6 STREP project, grant number IST-FP6-027253. In addition I am grateful for feedback from the wider research community, especially for the constructive comments I received from the reviewers of the IEEE Special Issue on Recommender Systems. I have also benefited greatly by attending the WWW-06, ECAI-06 and ACM MM-06 conferences.

Finally, I'd like to acknowledge the immesurable support I received from my family. My parents Louis and Lucy, my sister Leni, and my brother Stefanos have always given me love and encouragement. I dedicate this dissertation to them, for always seeing the best in me, even when they are experiencing the worst.

*To my family.*

# Chapter 1

# Introduction

The *Information Overload* problem is not a new one. Murray, [62] made the following observation in 1966: "*Every day, approximately 20 million words of technical information are recorded. A reader capable of reading 1000 words per minute would require 1.5 months, reading eight hours every day, to get through one day's output, and at the end of that period would have fallen 5.5 years behind in his reading*". The trend is exponential, and the problem is becoming ever more acute as the volume of available digital information, electronic resources, and on–line services increases. Given such an enormous corpus of data, ensuring that every piece of information that would be considered interesting by particular people is delivered to them (and not to others) becomes a problem.

Under the assumption that people are always aware of what it is they would be interested in, the aforementioned problem becomes a (non–trivial) search and retrieval task. This, however, is not always the case. It is human nature to constantly discover new interests, new challenges, and new forms of entertainment as facilitated by ones social environment. However, the identification of interesting information is not enough; information has to be prioritised, otherwise it could become overwhelming. This observation outlines a need for systems that are able to support this process by predicting how interesting unseen and unsought pieces of information will appear to a given person. Such systems are called Recommender Systems (RS).

In broad terms, RSs use machine learning techniques to process historical data about the interests of their users, encoded in user profiles. Once the algorithms used have been trained on user profiles, their output is used to compile a ranked list of all resources available for recommendation, based on each profile. The resources ranked at the top of each list are subsequently recommended to the corresponding user, as they are considered the most likely to be consumed next.

There has recently been a rapid increase in the commercial use of RS technologies, primarily by online retailers. Such systems appear attractive to retailers, since they can be used to identify any products from their catalogue that can be expected to appear interesting to a particular customer, thus increasing the amount of purchases made. This emphasis on increased sales is confirmed by Jim Bennet, of NetFlix, at his invited talk at Recommenders06: *"...Most of the time we don't actually have to sell them our movie, because they already know it, and just have to recognise it. So as long as we can pull those out, we are actually doing a pretty good job. ...Even though all the disks look the same, they don't cost the same for us to acquire."* [6]. NetFlix is an online DVD rental company that utilises a RS, typically dealing with billions of ratings and predictions. It is considered one of the most successful commercial RS deployments. However, the point being made is that the quality of a recommendation is measured by whether or not it leads to a rental, not by whether users are recommended movies which they like and may not have known about otherwise. Moreover, recommendations that lead to the rental of older, or not very popular movies are preferred since such films can be acquired relatively cheaply by NetFlix.

This view detracts from the notion of recommendation in two ways. First, people seek recommendations for things they have limited knowledge of, and use clever search and indexing techniques to remember things they were previously interested in. Secondly, the merit of a recommendation should be measured solely on the degree of user satisfaction, rather than by taking into account the relative profit of the recommending agent. Literature in the field provides support for such arguments. In [57], it is argued that the focus in RS applications should be shifted towards serving individuals instead of commercial sites. In a similar vein, [78] indicates that RS performance should really be measured by how effectively the system helps its users make decisions, rather than measuring how much profit it generates for a commercial website.

The recommendation framework proposed herein explores the opportunities that emerge by moving away from the paradigm of promoting product sales. In doing so, the set of resources available for recommendation is not dictated by the stock resources. Therefore, systems that enable the dynamic import of resources for recommendation can be developed. It is argued that a wider range of user interests can then be supported by such systems, by importing and recommending resources from multiple domains.

Further, user profiles in commercially deployed RS are typically compiled incrementally, by integrating user feedback on the recommendations they receive. At the same time, meaningful recommendations can only be made after a sufficient amount of feedback is provided by the users. Thus, it is assumed that once users have subscribed to the system, they will continue to assess recommended resources (even if they are not interested in them) until they have generated enough information to drive the system towards appropriate recommendations. However, end users can not be expected to have an intuition of how much information is enough. As such, they can easily become disappointed by the low initial quality of recommendations, and abandon the system.

To this end, user profiles are compiled automatically within the proposed framework, by unobtrusively recording information accessed or created by the users, independently of their interactions with the RS. As a large spectrum of user activities is being monitored, the resulting profiles are expected to contain sufficient information to capture user interests across multiple domains. Moreover, by semantically integrating elements in user profiles based on contextual attributes, users are able to select subsets of their profiles which they judge relevant to their current recommendation need, driving the system towards recommendations that are specific to it.

The next section first provides the main statement of this dissertation, followed by a discussion on the main aspects of the novel work carried out to develop a framework that enables the provision of recommendations from multiple domains.

## 1.1 Thesis Statement and Contributions

**A framework can be developed to enable Recommender Systems to consider multiple domains from which recommendations may be drawn. Moreover, as a consequence of their cross domain nature, such systems will provide recommendations that are of high quality, unexpected, and geared solely towards satisfying user needs.**

Figure 1.1 outlines the various contributions made to the *state–of–the–art* in support of this thesis:

**Integrating domain resources, expert preferences, and user profiles:** As stated above, resources for recommendation are imported to the system from external sources. Social networks that openly publish the profiles of their users are exploited to provide such resources. For example, Last.fm [48] exposes the music its users listen to, while del.ico.us [93] publishes the URLs bookmarked by each of its subscribers. As such, each source of information is considered an independent domain. The community of users for each social network is then considered to be a a set of domain experts, who publish their subjective evaluations of domain resources. The system is then required to assess which domain resources are most related to a set of domain independent profiling features. To do so, semantic relationships between expert preferences, domain resources, and profiling elements must first be identified. In the context of this work, Wikipedia was used in order to identify such relationships. Thus, a number of ways to obtain Wikipedia articles representative of domain resources or profiling elements are provided. Hyper–links between such articles are then considered as indicators of implicit semantic relationships. Such relationships are also assumed between the domain experts and the resources they evaluated. Moreover, the evaluative methodology used to assess

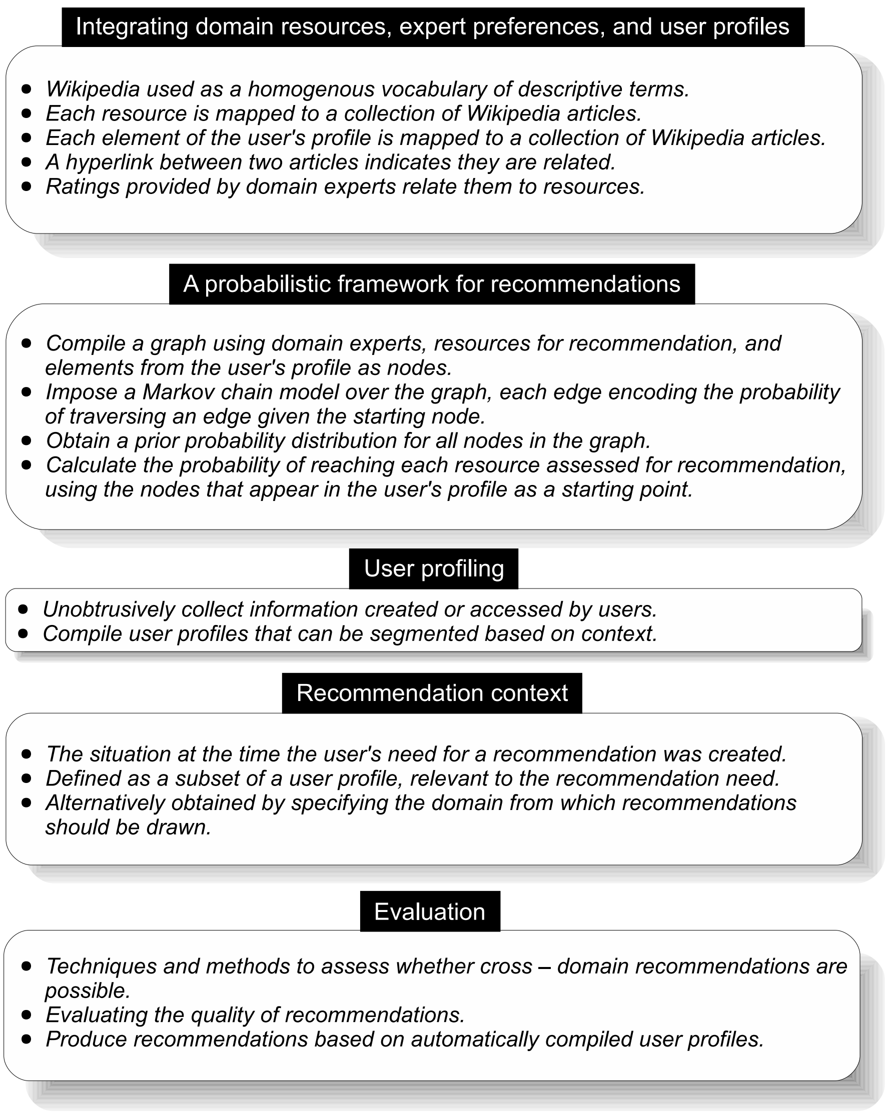

Integrating domain resources, expert preferences, and user profiles
Wikipedia used as a homogenous vocabulary of descriptive terms.
Each resource is mapped to a collection of Wikipedia articles.
Each element of the user's profile is mapped to a collection of Wikipedia articles.
A hyperlink between two articles indicates they are related.
Ratings provided by domain experts relate them to resources.
A probabilistic framework for recommendations
Compile a graph using domain experts, resources for recommendation, and elements from the user's profile as nodes.
Impose a Markov chain model over the graph, each edge encoding the probability of traversing an edge given the starting node.
Obtain a prior probability distribution for all nodes in the graph.
Calculate the probability of reaching each resource assessed for recommendation, using the nodes that appear in the user's profile as a starting point.
User profiling
Unobtrusively collect information created or accessed by users.
Compile user profiles that can be segmented based on context.
Recommendation context
The situation at the time the user's need for a recommendation was created.
Defined as a subset of a user profile, relevant to the recommendation need.
Alternatively obtained by specifying the domain from which recommendations should be drawn.
Evaluation
Techniques and methods to assess whether cross – domain recommendations are possible.
Evaluating the quality of recommendations.
Produce recommendations based on automatically compiled user profiles.

FIGURE 1.1: A visual representation of the contributions in this Thesis

the appropriateness of Wikipedia for this purpose can be generalised, providing an effective way to guide the selection of a corpus onto which to project domain resources and profiling features.

**A probabilistic framework for recommendations:** The result of the process described above can be naturally represented using a graph, with edges representing hyperlinks between Wikipedia articles. As such, the machine learning component of the framework is a graph–based, probabilistic algorithm. This was also motivated in part by the success of similar algorithms in estimating the significance of documents on the Web, by assessing its link structure (e.g. Google's PageRank [13]). A probability is then associated with the traversal of each edge, and a prior probability distribution over all nodes in the graph is obtained based on the data. Under this representation, the task of assessing resources for recommendation is equivalent to identifying the nodes in the graph that will most likely be reached by traversing the graph, using the set ofnodes representing profiling elements as a starting point.

**User profiling:** Having placed focus on e–commerce applications, Recommender Systems are typically designed under the assumption that an exhaustive index of resources for recommendation is available. Thus, user profiles only contain information pertaining to resources that appear in this index. We argue that this assumption limits the appropriateness of the recommendations that are produced, and also provides opportunities for the operators of such systems to promote certain resources, in order to achieve larger profits. Instead, this dissertation will show how any information accessed by a user can be automatically (and unobtrusively) compiled into a user profile, to obtain a more complete representation.

**Recommendation context:** The act of seeking a recommendation is always associated with a particular recommendation need. While this may be clear for the users of a RS, it is hard to capture automatically. However, the integration of profiling data based on contextual attributes, through the use of Semantic Web technologies, provides a variety of ways for users to intuitively select subsets of their profile that they judge relevant to the need they are currently experiencing. These context–specific subsets can then be used as the input to the algorithm to avoid recommendations that may be interesting to the user, but not useful in fulfilling their current need.

**Evaluation:** Experiments have been designed and carried out to evaluate each aspect of the framework presented above. As this is the first time (to the author's awareness) that recommendations across independent domains are evaluated on a large scale, it is of paramount importance to first assess the feasibility of such a task. To do so, the system's performance in producing recommendations from the same domain as the profiling features is evaluated. Once it has been established that the system behaves as expected for a single domain, a comparative study is carried out to assess changes in performance when profiling features and recommended resources appear in different domains. Moreover, we examine whether the profiling features collected through the monitoring of user activities are sufficient to capture user interests in multiple domains. This is carried out by assessing differences in the recommendation lists produced by the system, when profiling features from different sources of information are used. The final point of assessment lies in testing the assumption that restrictions placed on the resources that appear in user profiles based on contextual attributes associated with the resources, results in corresponding changes in the recommendation lists produced by the system.

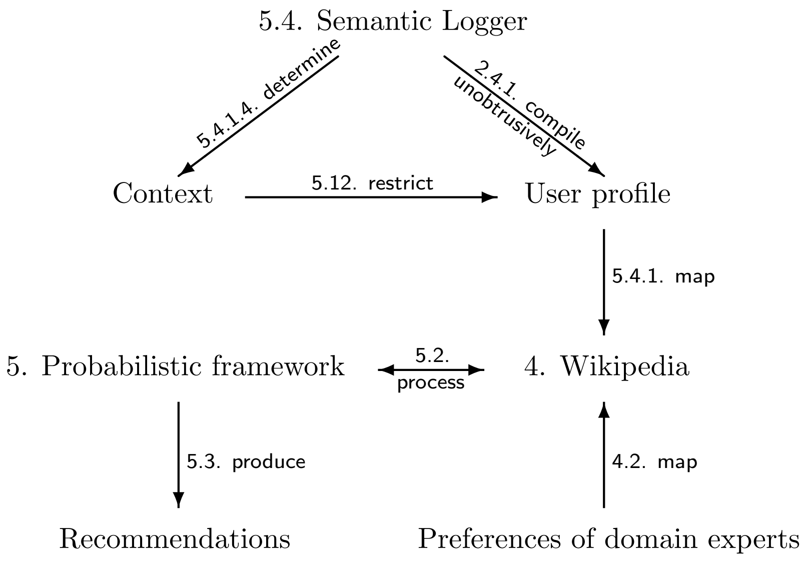


Figure 1.2: Conceptualisation of the interdependencies between the main matter of this dissertation. Nodes and edges provide navigation cues to relevant sections of the document.

## 1.2 Document Structure

The next chapter gives an overview of background literature. It first provides a review of the main RS design paradigms, identifying the prominent characteristics of each approach. In addition, the utility of technologies and data associated with the Semantic Web and Web 2.0 phenomena for recommendation purposes is considered. Following that is a discussion on a selection of seminal machine learning algorithms, along with metrics and methods used to evaluate the performance of such algorithms in Recommender Systems.

Chapter 3 first provides formal definitions for the concepts and functions that are used throughout the document. The terminology is then used to give a high–level description of the recommendation process. Finally the intuition behind the experiments carried out to assess each stage of the process is provided.

Figure 1.2 provides a visual interpretation of the interdependencies between the main matter of this dissertation. The shape of the figure resembles a line drawing of a house, on purpose. The foundations of the framework are in the effectiveness of the probabilistic model to capture the patterns that appear in the input. In turn, the model relies heavily on the link structure of Wikipedia to capture semantic relationships between heterogeneous concepts. The use of Wikipedia as a mediator between the various domain specific resource representations is provided first in Chapter 4. This is the case as it must first be established that the link structure of Wikipedia is sufficient in correctly representing relationships between resources, before these representations are used to produce recommendations in Chapter 5.

As such, Chapter 4 presents and evaluates three different modes of obtaining Wikipedia articles representative of domain resources. Each mode applies to a different set of domain characteristics, providing a comprehensive toolset that is expected to be applicable to the majority of available domains.

The 'roof' of the structure in Figure 1.2 is the Semantic Logger, the system responsible for the unobtrusive collection and integration of information created or accessed by users. While the roof is not particularly important to the structural integrity of a building, it is essential for maintaining a constant environment inside it. Similarly, while recommendations could theoretically be provided for each possible combination of user input and recommendation domain, it is argued that by logging as much information about user activities as possible, the probability of not capturing some aspect of user interest is minimised.

Glyphs of the same shape as the figure are provided in the margin at the beginning of each main chapter, and serve to provide a visual reminder of where the chapter fits in, with respect to the framework.

The document concludes in Chapter 6 by providing a summary of experimental results with respect to the central thesis of the dissertation. Finally, open research questions that have emerged from the various aspects of this work are articulated, and directions for future work outlined.

## 1.3 Publications

During the development of this dissertation the following work has been peer–reviewed and published:

- Antonis Loizou and Srinandan Dasmahapatra. **Recommender Systems for the Semantic Web.** In *ECAI 2006 Recommender Systems Workshop*, August 2006.
- Mischa M Tuffield, Antonis Loizou, David Dupplaw, Srinandan Dasmahapatra, Paul H. Lewis, David E. Millard, and Nigel R. Shadbolt. **The Semantic Logger: Supporting Service Building from Personal Context.** In *Capture, Archival and Retrieval of Personal Experiences (CARPE) Workshop at ACM Multimedia*, October 2006.
- David Dupplaw, Madalina Croitoru, Antonis Loizou, Srinandan Dasmahapatra, Paul Lewis, Mischa Tuffield, and Liang Xiao. **Multimedia Markup Tools for OpenKnowledge.** In *1st Workshop on Multimedia Annotation and Retrieval enabled by Shared Ontologies*, December 2007.

# Chapter 2

# A critical review of background literature

## 2.1 Recommender Systems

This section provides an overview of the main Recommender System paradigms and the intuition behind their inner workings. However, it is not intended to be a systematic analysis of the algorithms used, rigorously pinpointing the points of failure; for this, the interested reader is pointed to [2]. This is the case since the scope of this dissertation is not to invent the perfect recommendation algorithm, but instead to identify a process where any (internet accessible) resource may be recommended, if appropriate.

### 2.1.1 Content based filtering

Under the assumption that users will like similar items to the ones they liked before, content based filtering (CBF) systems record descriptive features of items and try to identify the most similar ones in the catalogue, [5, 65]. This type of architecture is greatly influenced by research in data mining, classification and machine learning in general, since the recommendation problem is reduced to the question "Is this item sufficiently similar to those in the training set?".

A vector is constructed for each item containing values for the descriptive features of each item and is considered a point in a multidimensional space. Inter–item similarity is assessed by evaluating the distance between such points. This reduction however is not always correct, since the most similar items to those already seen by a user can often be of little interest, if the original recommendation need has already been satisfied. Content–based systems fail to acknowledge a vital difference between classifiers and recommenders; recommenders should return resources that are not only relevant, but also useful to the user at the time of recommending.

CBF is usually applied to textual domains, such as news items [10], and are typically implemented with variants of the Term Frequency – Inverse Document Frequency (TF–IDF) algorithm [72]. It is much more difficult, and computationally expensive to extract content based descriptors of multimedia items in order to assess their similarity, but with the recent increase in the amount of rich descriptive metadata available for such resources, the problem is becoming less acute. One of the strengths of this approach is the fact that a prediction score can be obtained for every resource–user pair, provided that the user has a non–empty profile. However, CBF systems often fail to produce recommendations that have different, but related content.

### 2.1.2 Collaborative filtering

Systems that apply Collaborative Filtering (CF) assume that users will be interested in items that users similar to them have rated highly. This paradigm has been originally put forward by two independently developed systems, Ringo/Firefly [81], and GroupLens [69], in 1994 and 1995 respectively. The entusiasm associated with the deployement of these seminal systems, together with the large number of deployed RS implementations that use this strategy [49, 11, 67, 95, 49, 6, 48] indicates that the assumption made generalises well. The active user is first matched to the group of most similar users using a similarity metric and a clustering algorithm – typically $k$ Nearest Neighbours. Then, items seen by the group but not by the active user are identified. The predicted rating for each unseen item is then computed by aggregating the group's ratings. This is typically weighted by the number of group members that have accessed a particular item and the variance between ratings, effectively biasing the process towards items that more people in the group have unanimously rated highly. Finally, the items with the highest predicted rating are recommended. The approach has been shown to produce unexpected, but high quality recommendations, as it ultimately depends on aggregating opinions expressed by humans.

The main point of failure in this architecture lies in choosing the correct cluster for each individual user. Due to the level of sparsity in the datasets RS typically deal with, users can appear equally similar to any other user if the similarity metrics used are not sensitive enough, [76]. As such, the user is merely provided with recommendations for the items most popular with a group of 'randomly' selected users, and there could be a potentially high degree of disagreement among members of this group. The reverse effect is also present: items can only be recommended after being rated by a sufficient number of users. Problems associated with the lack of information regarding newly subscribed users or new items available for recommendation are commonly referred to as "cold-start" issues [79]. Scalability is another issue that needs to be addressed when collaborative filtering is applied. For a dataset of $m$ users and $n$ items the algorithm runs in $O(m^2n)$ time, as it requires the comparison of $m(m-1)$ pairs of $n$–dimensional vectors. Dimensionality reduction techniques such as Singular Valued Decomposition [74] are often applied to reduce the effect of data sparsity and improve performance, but are computationally expensive and do not always resolve the issue in cases of extreme sparsity.

There are three tuning parameters for CF systems. The similarity metric, neighbourhood size, and the size of the recommendation list to be produced. Common choices for the similarity metric include Pearson's correlation coefficient [66], the Spearman correlation [82], vector cosine distance [71], mean–squared differences [16], and Euclidean–based distance measures. The neighbourhood size will effectively dictate how personalised the recommendations produced will be; a small number of neighbours can lead to spurious recommendations and very large neighbourhoods will result in recommending popular items only. Finally, the number of recommendations produced is a vital factor as it affects the perception the users have about the system. Small lists could miss out important resources, while large ones can prove useless and confusing to the users. Note that the upper bound on the number of recommendations that can be provided is given by the number of distinct resources rated by the user's neighbourhood.

Item–based collaborative filtering was introduced in [75], expressing the intuition that there are relationships between the resources themselves, rather than between the users that consume them. As such, neighbourhoods of similar items are created and used to produce recommendations. Similarity between items is calculated based on which users have rated them – each item is considered to be a vector of ratings provided by the users, and the distance between such vectors can be evaluated using a variety of metrics. Experiments carried out in [75] show that this approach is more efficient than user–based CF, running in $O(n^2)$ (where $n$ is the number of items), while retaining the high quality of recommendations in terms of predictive accuracy. The adoption of this method by [49] is seen as further testimony of its merits over its user–based counterpart.

### 2.1.3 Demographic filtering

Demographic filtering (DF) shares the view expressed by CF, in that similar users are expected to share the same interests. However, this approach tries to tackle the recommendation problem from a different, somewhat more general perspective. Instead of using the ratings provided by users to compile profiles, users are asked to provide demographic information such as their age, interest in sports, favourite TV programs, and purchasing history, among others. They are then compared to pre–compiled clusters of the general population, indexed by the same characteristics. Sets of resources available for recommendation are matched *a priori* with such clusters. Once the most similar cluster has been obtained, the resources associated with it are recommended.

Demographic filtering was first introduced by [46]. They used the classification of more than 40 000 people from the USA into 62 demographic clusters based on the products they purchase, carried out by [92]. Using this dataset to train against, they were able to report encouraging results by matching new users to population clusters, and then recommending products explicitly associated with members of that cluster. The observation that profiling features independent of the recommendation domain can drive a RS is made here for the first time, and is seen as the earliest pre–cursor to the idea of carrying out cross–domain recommendations. However, its dependence on previously collected data, and the requirement of explicitly mapping the resources available for recommendation to population clusters render it rather cumbersome and labour intensive. Furthermore, it is possible that the quality of recommendations is negatively affected by the over–generalisation of user interests; it is highly improbable that someone interested in sports will follow each existing sport, or that a fan of horror movies would enjoy every film of that genre. Because of such limitations, demographic filtering is typically used as one of several components in hybrid RS, discussed later in this chapter.

### 2.1.4 Knowledge based systems

Knowledge based systems use content–based information together with explicit background domain knowledge and user models to carry out reasoning, and produce recommendations. By doing so they have obtained unique qualities that separate them from the approaches discussed earlier. This section gives a short overview of such systems.

#### 2.1.4.1 Rule filters

Rule filters were used in the first RS, Tapestry [30], and either require a user to explicitly formulate rules to filter out bad recommendations (eg. "I never watch musicals"), or try to infer such rules based on the user's history. A number of drawbacks are apparent in this architecture. Users find defining rules in a formal language awkward and cannot be expected to formulate good quality rules. On the other hand, the automatic approach can be very complicated and can produce rules that do not reflect user preferences but happen to produce good results in the training phase by chance. This is a problem because as the user accesses more items such rules can become conflicting and need to be reassessed. Carrying this out is the object of ongoing research [12]. Moreover, to be able to define rules to constrain which items can be recommended, users are required to be aware of exactly what they'd like to be recommended to them which detracts somewhat from the notion of a recommendation.

#### 2.1.4.2 Case–Based Reasoning and Conversational RS

Case–Based Reasoning (CBR) is traditionally thought of as a problem solving methodology. The system stores previous problem solving episodes as 'cases'. When a new problem is posed, the system looks for similar ones that have been solved in the past and suggests their solutions for the new one. For recommendation purposes, user models are analogous to problems and the potential solutions to recommendations [88, 59]. CBR recommender systems create an interactive recommendation experience, gathering information at each step of the process to refine and further customize the recommendation list produced. Owing to this fact such systems have also been dubbed 'Conversational Recommender Systems'. Two distinct methodologies are typically applied when developing such systems. The first one (eg. [70]) involves iterating over questions that narrow down the space of items that can satisfy the constraints, using (among other metrics) entropy to decide on which question to ask next. Alternatively (eg. [52]), lists of resources are presented to the user, who selects the ones that appear the most interesting. The system uses this selection to infer rules, and presents a new list. This process continues until the user chooses to consume a resource. Systems of the latter type are found to promote domain exploration and resource discovery by users. As with rule filters, conflicting requirements may lead to a situation where no resource can satisfy the constraints, and constraint relaxation has to be carried out.

#### 2.1.4.3 Ontology based systems

Systems in this category make use of an underlying ontology to describe both the users of the system and items to be recommended. Recommendations are provided via assessing the similarity between instances associated with the user and all other instances in the system's knowledge base, by applying graph-based edge expansion heuristics [3]. In essence, the various relationships between users and resources (eg. user A knows user B, user B likes resource 1, user A created resource 2, resource 1 is a revision of resource 3, etc.) are weighed and the graph is traversed to find the most related resources. Such an approach also enables the user to visualise and amend their profile in order to reflect their preferences more accurately, since it is represented as a graph. One such system, Foxtrot, [56] has empirically been shown to outperform other strategies, such as CF and CB recommending, however as with any knowledge based system knowledge acquisition poses a problem. This is typically dealt with by employing (usually labour intensive) bootstrapping techniques. The power of the system lies in exploiting semantic relationships between users and resources, and has provided some motivation for deploying the graph based approach presented in Chapter 5.

### 2.1.5 Context dependent systems

The architectures mentioned above (with the exception of knowledge based methods) all use a flat matrix representation of the problem domain where rows correspond to users and columns to items. As mentioned before, these tend to be vast and very sparse, thus adding to the problems of defining efficient similarity metrics and computational complexity. Adomavicius *et al.*[1] propose that contextual dimensions (such as time) should be added to this representation, so that at the time of recommendation a relatively dense 'slice' of this space is used, selected based on the current context. Take for example a system that provides movie recommendations, with the added contextual dimension being the 'day of the week'. Any recommendations that will be made on Fridays will only take into account movies that have been seen by other users on a Friday. This space is clearly smaller, and all users and resources contained within it have at least one rating. They show that such an approach is beneficial in most cases and have defined a heuristic to evaluate a priori whether the multidimensional approach is likely to outperform the traditional flat representation. Where it is not the system falls back to a conventional recommendation scheme.

### 2.1.6 Hybrid systems

Hybrid systems combine two or more recommendation techniques in order to improve the quality of recommendations they provide [5, 64, 1, 7]. A comprehensive analysis of such systems can be found in [14]. In summary, hybrid RSs can be split in three broad categories:

- Weighted hybrid RS assign weights to each strategy used, and aggregate the results from each one to compute the predicted rating for a recommendation.
- Mixed hybrids will provide any recommendations above a confidence threshold from each scheme used.
- Hybrids where a single strategy is chosen each time, based on a heuristic to identify the one likely to perform best.

It should be noted that the first two design paradigms for Hybrid Systems introduce large increases in the computational requirements of the systems developed, since a number of distinct recommendation techniques are carried out. Conversely, accurately predicting the performance of the recommendation algorithm is no simple task. Various heuristics (such as the sparsity index of the entire space, the number of available ratings for the active user, etc.) are commonly used to predict the algorithm's performance, incurring minimal, but observable losses in terms of predictive accuracy.

### 2.1.7 Cross–domain recommendations

Performing recommendations across different domains is a very new area of research, and the number of publications addressing such issues very small. In broad terms, developments in this area can be split into the following categories:

**Compiling user profiles appropriate for cross–domain recommendations:** The authors of [32] indicate that in order for cross–domain recommendations to be made, the question of integrating domain specific profiles, with generic, domain–independent ones needs to be addressed. They provide a formalisation for such domain–independent profiles called Smart User Models ($SUM$s). A $SUM$ is a collection of attribute–value pairs, that characterise the user along the axes of objective ($O$), subjective ($S$), and emotional ($E$) characteristics. Each set of characteristics forms a component in the user model, $U^O$, $U^S$, and $U^E$ respectively, giving $SUM = \{U^O, U^S, U^E\}$. In addition, a User Model is compiled for each domain, $UM_i = \{AD_i, AI_i, AU_i\}$, where $AD_i$ is the set of domain characteristics for domain $i$, $AI_i$ the domain specific set of user interests, and $AU_i$ a set of socio–demographic features required to be specified by the domain. A weighted graph $G(SUM, UM_i)$ is used to establish a relationship between the $SUM$ and $UM_i$. The graph connects the features in the two models, assigning weights proportional to the relatedness of $SUM$ features to their $UM$ counterparts. The work consists of a formal description of the models, and does not provide a clear method to produce recommendations using $SUM$ user profiles.

**Cross–domain profiling for single domain recommendations:** Influenced by demographic filtering, [45], deploys a *web user agent* to carry out the task of learning user interests through monitoring web usage in multiple domains. The information is then exploited by another agent to produce recommendations by re–ranking web links, and automatically delivering interesting web pages. The unobtrusive nature of the profiling component renders it similar to the approach described in this dissertation. However, within the context of this work, monitoring web usage is only one aspect of user profiles, as will be presented in Section 5.4. Moreover, while the argument that various recommended Web pages will belong to different domains is made, this distinction is unclear.

**Recommendations from multiple domains, using cross–domain profiling information:** Berkovsky et al., [8], propose that cross domain recommendation techniques can be exploited to alleviate the negative effects of data sparsity on CF systems. They identify four modes of cross–domain mediation between CF Recommender Systems:

1. Exchange and aggregation of user profiles.
2. Exchange of user neighbourhoods.

3. Communicating degrees of similarity between users.
4. Exchange of recommendations.

In order for such modes of mediation to become possible, the authors stress that remote systems must exploit the same CF recommendation technique as the target system, in other application domains. In order to evaluate their work, the authors segmented the EachMovie dataset, [53] by genre. Each genre was then considered an independent domain.

The first type of mediation is particularly useful to newly deployed systems. Even if the first users of such systems are willing to provide ratings, there is limited information within the system to compare these profiles with. In this case, user profiles may be imported from a remote system to rectify the situation, provided that the exporting system indexes a subset of the available resources.

In the second instance, the remote system is asked to export the neighbourhood obtained for a user profile provided by the target system. Such a situation may arise when the active user is not similar enough to any cluster of users within the local RS. Provided the neighbourhood has rated resources available to the target system (or the same users are present), the ratings can be processed to make recommendations.

Whenever similarity between users cannot be computed due to the level of data sparsity, a remote RS may be able to provide such a value based on its own index of resources. This may be preferred to the previous type of mediation, since ratings from the local RS can still be used for recommendation purposes.

Finally, if recommendations cannot be made locally with high enough confidence, the active user could be injected into the remote system, which would then produce recommendations and export them back to the target RS.

This approach, however, is only applicable if the systems involved index some resources in common, or share a number of subscribers. Further, the recommendation process would have to be identical in order to exchange similarity values and user neighbourhoods. Due to these limitations the applicability of this approach is severely restricted.

### 2.1.8 Well known pitfalls in deploying RS

- The 'cold start' problems

  In CF–based recommending, the similarity between user profiles is assessed to predict ratings for unseen items. The shortcoming of such a method rests in its assumption that active users will respond positively to unseen items rated highly by similar users [67]. As most users are not inclined to rate previously seen items, only a few items will receive ratings. This limited data – the 'cold start' problem

– renders similarity metrics not sensitive enough to distinguish between users, particularly new ones introduced to the system, [79]. Hence, the most highly rated items from anyone are recommended. The reverse effect is also present, i.e. a newly imported item can not be recommended until it receives sufficient ratings.

- The most similar items are not always good recommendations

  Content based filtering (CBF) approaches index the items of possible interest in terms of a set of automatically derived descriptive features, and unseen items with similar attributes to those rated highly by the user are recommended. A drawback of the CBF method is that it recommends items interchangeable with those that have previously received high ratings, by virtue of its focus on items' features, ignoring potential user requirements. As such, for a system to be able to avoid such issues, equivalence (or subsumption) between items, under particular contexts, needs to be evaluated.

- Shifts and temporal cycles of user interests

  Most conventional RS architectures do not model for shifts of the user's interest over time, since all ratings provided by a user have an equal bearing on producing recommendations. To clarify this point consider the following scenario: a user $X$ has provided high ratings only for items in some set $A$, however s/he is now only interested in items from another set, $B$. A conventional RS will not be able to recommend items from set $B$ until enough ratings are provided for items in $B$, in order for them to dominate in the clustering and selection processes that make up the recommendation algorithm. It may even be the case that $X$ is only interested in items from $A$ during the weekend, while only items from $B$ are of interest during the week. This means that a system should not become stable, and that the classification of the same items to different classes, at different times, may be deemed correct, something that would have to be taken into account in a machine learning context. To accommodate this requirement of preference time dependence, conventional architectures recompute their user clusters periodically, effectively choosing a different training set every time. This can aggravate problems caused by data sparsity, and important modeling decisions about transitions between user needs have to be addressed.

- Recommendations made independently of context

  An interesting resource does not necessarily make a good recommendation every time. Typically, recommender algorithms do a good job at identifying resources that are similar (or relevant) to those already consumed by users. In most cases however, they fail to capture the reason the user is seeking recommendations. As such, the recommended resources may fail to fulfill the user's recommendation need, while being interesting at the same time.

- Only items described in one pre-specified representation are considered

Since the focus in RS applications has been to enable organisations to suggest appropriate items from their catalog to customers, not much effort has been put into learning user preferences based on the items they already have in their possession, regardless of their origin. However, a good sales assistant in a clothing shop will first look at what the customer is wearing before making suggestions.

- Potential biasing effects

  Following on from the previous point, the fact that the provider of the recommendation service is typically the vendor of the resources available for recommendation introduces added considerations. Since the vendor stands to profit from the users of the RS and resources have varying profit margins, it is highly conceivable that they will introduce bias towards producing recommendations that if consumed would maximise profit for the vendor. Further, it can be expected that in situations where the system cannot make any recommendations with high confidence, popular items are recommended in the hope of a sale. Both these phenomena have been observed, and are seen as diverting focus away from satisfying user needs.

## 2.2 Evaluating recommender systems

### 2.2.1 Metrics and methods used in evaluating predictive accuracy

This section presents some widely used methods and metrics to assess the predictive accuracy of a recommender algorithm, and provides a discussion on their appropriateness for the task.

#### 2.2.1.1 Leave–$n$–out

As mentioned earlier, the leave–$n$–out methodology is typically used to test the accuracy of classification algorithms in Machine Learning. Under this scheme a portion of the data is hidden, and the algorithms performance in reconstructing it is evaluated. The metrics discussed below address the question of how this evaluation can be performed. However, this methodology is less than ideal when applied to the Recommender System domain. The sparsity of user profiles is a well identified problem - by removing a substantial portion of the ratings available, similarity values between users may be severely affected. In addition, there is the possibility that the system will produce recommendations that are better suited for the user than the withheld ones. In other words, there may be resources that users would have expressed interest in, and even more so than for resources that they have already consumed, had they known of their existence. After all, RS are designed to do just that. Evaluation approaches based on leave–$n$–out will not only fail to capture such successful recommendations, but will penalise systems that provide them. In order to minimise such effects, [55] proposes that only a single rating should removed from a small subset of user profiles (approx. 10%) during the training phase.

#### 2.2.1.2 Precision, Recall and the F1 measure

Precision and recall, [29], are typically used in Information Retrieval to assess the quality of query results using labeled test data. Precision is defined as the number of positive results retrieved by a search, divided by the number of search results. In contrast recall is the number of positive results retrieved by the search divided by the total number of positive results in the corpus.

Assuming that a leave–*n*–out methodology is used, Precision and Recall can be expressed in the following way, in terms of Recommender Systems. Precision measures the portion of the resources recommended to a particular user that also appear in the list of withheld items, while Recall measures the portion of withheld resources that are recommended. These definitions do not capture the recommendation task well, as positive results are restricted to those removed from the original profile. Clearly, if that was the case (i.e. that users are not interested in resources they have not already rated) there would be no need for deploying a recommender system. In addition, most recommendation techniques will produce confidence values associated with each resource available for recommendation, and the final list of recommendations is obtained by applying a threshold. As such, there can be discrepancies in calculating Precision if, for example, the set of withheld items is larger than that of the recommendation list.

The two metrics are complimentary in nature and can be combined together. Their combination in a single metric is expressed by the F1 measure, [90]:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.1}$$

#### 2.2.1.3 Mean Squared Error

Mean Squared Error (MSE), along with Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) are widely used loss functions in statistics. Such functions measure the amount by which an estimator differs from the true value of the quantity being estimated. MSE was first introduced by Carl Friedrich Gauss in 1809 [26]. The metric gives the average square difference between the estimator and the true value, across a number of cases. The relationship between MSE and RMSE is similar to that of variance and standard deviation. MAE on the other hand considers the absolute difference between the estimator and the true value rather than its square.

When such metrics are applied in evaluating the performance of Recommender Systems they become rather crude, as they assess the ability of the algorithm to reproduce the exact ratings assigned to resources by users, rather than values that will reproduce the ordering of resources in terms of user preferences.

#### 2.2.1.4 Kendall's $\tau$

Kendall's correlation coefficient, $\tau$, [43] is a widely used statistic to compare two different rankings of the same variables. The $\tau$ value is obtained by dividing the difference between the number of concordant ($n_c$) and discordant ($n_d$) pairs in the ranked lists, by the total number of pairs. A concordant pair is defined as two variables that appear in the same order in both rankings, while otherwise the pair is considered discordant. More formally:

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)} \tag{2.2}$$

Since $\tau$ is concerned with comparing lists of variables, rather than individual elements, it can be applied more naturally in evaluating RS. A possible evaluation scenario using this metric would be to compare the ranking of withheld resources based on the original ratings provided by the users, to the ranking of the same resources based on the confidence values computed by the system.

#### 2.2.1.5 Breese's Half–life metric

Breese's Half–life metric, [42], also referred to as the Breese score, is of particular interest, since it has been designed with Recommender Systems in mind. The metric attempts to assess the utility of a list of recommendations to a user, postulating that the lower an interesting resource appear in the recommendation list, the less likely it is that the user will see it. It is implemented as an exponential decay function, where the half–life value, $a$, is defined *a priori* as the rank of the resource in a recommendation list that has a 50% chance of being seen by the user. The Breese score is given by:

$$R_a = \sum_j \frac{max(u_{a,j} - d, 0)}{2^{(j-1)/(a-1)}} \tag{2.3}$$

where $u_{a,j}$ is the rating provided by user $u$ for the resource with rank $j$ in the recommendation list and $d$ the 'neutral' rating. $d$ can be a default value for all users (e.g. 3 on a 5–point scale) or the user's average rating.

### 2.2.2 Human Recommender Interaction (HRI) theory

Human Recommender Interaction, [55], provides a theoretical grounding for analysing the utility of a RS from a user–centric perspective. It is a descriptive theory, intended to provide an insight into how users perceive the recommendations provided to them. Such

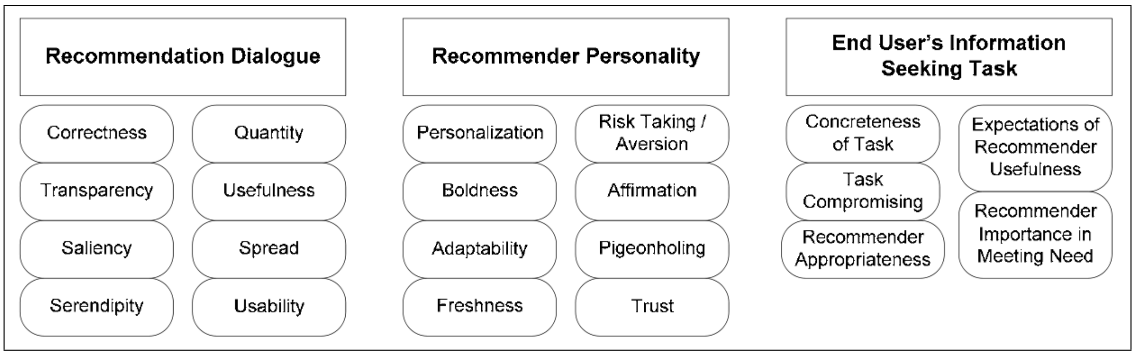


FIGURE 2.1: The *Pillars* (□) and *Aspects*(⬭) of HRI, as presented in [55].

insights are considered a requirement for improving the quality of the recommendations made by automated systems. It is presented as a framework and a common language to use when describing the kinds of recommendation lists an algorithm produces. However, when HRI is added to a larger process model it can become constructive and used as a tool to analyse and redesign RS. The words in this language are called *Aspects*, and are organised in three *Pillars*, as shown in Figure 2.1.

The first *Pillar*, Recommendation Dialogue, contains eight *Aspects* considered to be pertaining solely to tuning the Recommender System during a single recommendation session. It is focused on the run–time interaction between the user and the system, where the aspects are assigned significance values. The pillar includes:

Correctness: A correct recommendation is one judged as relevant and of high quality by the user.

Transparency: A transparent recommendation is one for which the user understands the reason it was recommended.

Saliency: Salient recommendations stand out in a recommendation list. They are vivid, unexpected, notable, conspicuous and prominent but not necessarily relevant. Thus salient recommendations generate strong positive or negative emotional responses from users.

Serendipity: Serendipity means making a fortunate discovery by accident. A serendipitous recommendation is one that the user did not expect to receive but considers interesting. Most likely, but not necessarily, a serendipitous recommendation will also be salient. At the same time a list of salient recommendations could prevent a serendipitous one from standing out.

Quantity: The amount of recommendations provided by the system to a user. The trade–off between providing a small list of recommendations that may omit good quality recommendations and a larger one which may prove difficult to sort through is addressed here.

Usefulness: If a given recommendation is consumed by the user then it is considered useful. Note that under particular circumstances a correct recommendation may not be useful. The author gives the example of seeking movie recommendations to take a 9–year–old to the cinema. In such a case correct recommendations for the user may be unsuitable for the child, and thus less useful. Better recommendations may be achieved by restricting the user's profile to resources that are also relevant to the child.

Spread: Spread is a measure of how well the recommendation list covers the range of resources available to the system. Thus, large list of very similar items has lower spread than a smaller, more diverse list.

Usability: When the user understands how to use the RS to fulfill their information need, and the system responds as the user expects it should, we say that the system is usable.

The Recommender Personality *Pillar* is the overall impression created by the user about the RS, over a period of time. It is concerned with more permanent characteristics of the system, which can not be tuned at run–time:

Personalisation: Personalisation is a measure of how unique recommendation lists are for different users.

Boldness: A recommender is considered bold if many of the recommendations are for resources likely to receive 'extreme' ratings (either very high, or very low).

Adaptability: Adaptability measures how well changes in a user's profile propagate through the system to cause changes in the recommendations produced.

Freshness: Fresh recommendations concern resources that have neither been rated by the user nor appeared in previous recommendation lists.

Risk Taking/Aversion: A risk taking recommender is one biased towards producing recommendations for obscure and under–represented resources. It differs from boldness in that a risk averting recommender may be bold by strongly recommending a particularly popular item.

Affirmation: An affirming recommender would recommend items it expects the user to know and have an opinion about. Such systems may not generate very useful or personalised recommendations, but are focused on establishing rapport with its users.

Pigeonholing: If very similar recommendation lists are produced for a user over time, the system is said to exhibit pigeonholing behaviour. In contrast to spread, which is concerned with a single recommendation list, this aspect describes the (lack of) breadth of recommended resources over an extended period.

Trust: The trust aspect refers to how well a system earns the trust of its users on various levels: that the system will provide reasonable and accurate recommendations; that they will not be deceived by the system; that the system will behave similarly in future interactions; that the recommendations will improve the more information the users provide; that their data is kept private. It is considered an extremely important factor influencing the user's decision to continue using the system.

The final *Pillar*, End User's Information Seeking Task is different in nature from the aforementioned ones. Before, *Aspects* provided a language to express desired properties in recommendation lists. Here, they refer to the types of task that may drive a user to a RS, thus acting as a reference point for the system developers in choosing which tasks to support. The five *Aspects* contained in this last *Pillar* are:

Concreteness of Task: The user's ability to clearly express the task they are faced with.

Task Compromising: The willingness by users to modify (or compromise) a concrete task for which they seek recommendations, in the light of the recommendations made to them.

Recommender Appropriateness: A RS will not only be judged by its users on the correctness of the recommendation lists generated, but also on its appropriateness in fulfilling the user's need. While correctness and appropriateness are related, this aspect describes whether or not the recommender can help with a particular task.

Expectations of Recommender Usefulness: Since the need to obtain recommendations exists before a user interacts with a RS, it is impossible to state that there are no expectations. The system has a responsibility to make its capabilities clear to its users so that expectations can be reasonably managed.

Recommender Importance in Meeting Need: Users may use many different sources of information to fulfill their need. RS could be used either as a central information source to find information of interest, or as a secondary one to validate and verify information gathered from other sources. As this varies, the importance of other aspects, such as boldness, risk taking/averting behaviour, saliency, and spread will vary accordingly.

#### 2.2.2.1 The curse of accuracy

A very important observation is made in the introduction of [55]: the evaluation of RS has been focused solely on testing the predictive accuracy of the recommendations made.

A variety of metrics have been designed to measure accuracy, such as precision, recall and the F1 measure [73], Receiver–Operating–Characteristic (ROC)[24] and Customer ROC(CROC)[80] curves, Breese's half–life metric [42], amongst others. This emphasis on accuracy is seen as a consequence of the fact that research on Recommender Systems has strong roots in the Machine Learning and Classification fields. However, there is a clear conceptual difference between a classifier and a recommender. A classifier used for recommendations would find resources that the user is likely to rate next. These may have high ratability, but there is no guarantee that they will be appropriate at any given point in time, or be relevant to the reason the user needs recommendations. While the importance of accuracy to a RS can not be undermined, the author of [55] identifies problems that arise when it is made the sole focus of evaluating such systems.

At a high level, accuracy centric analysis views the recommendation process in the following way. The algorithm receives an input from each user in the form of ratings – mappings between resources and values that represent how interesting those resources are to the user. The algorithm performs a computation to predict how the user will rate every remaining resource, and returns a recommendation list that contains the items with the highest predicted ratings. In this setting each request sent to the system is considered independent of all others. This may hold for the algorithm itself, but not for the user. To illustrate, consider a situation where the system recommends a list of resources that the user knows to be of little interest. The user may then decide to ignore these, instead of providing (low) ratings for them. As such, the system is stuck in a loop, recommending the same resources every time. Metrics that judge independent events are bound to miss the temporal aspect of the recommendation process. Such issues could be avoided by assuming lower than average ratings for resources that were recommended, but not consumed.

Moreover, users see each recommendation in the context of the rest of the recommendation list they receive. Thus, independently accurate recommendations may result in a bad recommendation list. The example given is of a book RS, which correctly predicts that some user will be highly interested in Tolkien's *Lord of the Rings*. If the list provided contains 10 different versions of this book, should that be considered a good recommendation list? This problem outlines the need for evaluating recommendation lists as a single entity, rather than each individual recommendation.

Another factor to consider is the methodology used in carrying out evaluations of recommender algorithms – leave $n$ out [42]. Essentially ratings data is split into train and test sets ($n$ refers to the size of the test set). The algorithm will then learn the data in the training set and attempt to reproduce the data in the test set. Its performance is then judged based on the overlap of the original test set with the output of the recommender. Viewed from the user's perspective, such recommendations are useless. Even if the algorithm correctly ranks the resources that a user has already seen, it has proposed

| HRI | | Metrics for assessing recommendation lists | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Popularity | Ratability | Intra–List Similarity | Authority | Personalisation | Coverage, Rank, Adjusted Rank | Boldness | Adaptability |
| Recommendation Dialogue | Correctness | | | − | | | + | | + |
| | Transparency | + | | | + | − | | | |
| | Saliency | | − | | | + | | + | |
| | Serendipity | − | | − | − | + | − | | + |
| | Quantity | | | | | | + | | |
| | Usefulness | | + | | | | + | | + |
| | Spread | | | − | | + | + | | |
| | Usability | | | | | | | | |
| Recommender Personality | Personalisation | − | | | − | + | | | + |
| | Boldness | − | | | − | | | + | |
| | Adaptability | | | − | | + | | | + |
| | Freshness | | | | | | | | |
| | Risk Taking/Aversion | − | | | − | | − | + | |
| | Affirmation | + | + | | + | | | | |
| | Pigeonholing | | | + | | − | | | − |
| | Trust | | + | | | | + | − | |

Table 2.1: Mappings between HRI *Aspects* and performance metrics, derived by [55]. + indicates a positive correlation between a metric and an *Aspect* (i.e. scoring high on the metric translates to the *Aspect* being a prominent characteristic of the system), while - indicates negative correlations.

nothing new, since by doing so it would be penalised by accuracy driven performance measures.

In the light of these observations the author goes on to provide a list of metrics, designed specifically to support the evaluation of RS. These are described below, and their relationship to the *Aspects* of HRI detailed earlier, are provided in Table 2.1.

Popularity: Popularity is a measure of how widely known a resource is. Since the work is concerned with recommending academic papers, the following definition is given:

$$popularity(j) = \frac{citation\_count(j)}{years(j)}, \tag{2.4}$$

where *citation_count* is the number of citations paper $j$ received, and $years(j)$ the number of years since its publication. This expresses the intuition that popular papers are cited by a large number of other papers soon after they are published.

Ratability: Ratability is the probability that a user expresses an opinion on a given resource. A naive Bayes classifier is proposed as a measure of ratability, assessing which resource a user is most likely to rate next, given his current profile.

Intra–List Similarity (ILSM): The ILSM of a recommendation list is based on the pair–wise similarity of all items in the list, and requires the use of an external similarity measure. For a recommendation list $L$ of length $n$ and similarity metric $w(\cdot\ ,\ \cdot)$, the intra–list similarity is given by:

$$ILSM(L) = \frac{\sum\limits_{x \in L} \sum\limits_{y \in L, x \neq y} w(x, y)}{n(n-1)} \tag{2.5}$$

In [96] this metric was used to identify recommendation lists that contained highly similar resources. The lists were then modified using a topic diversification algorithm, that effectively replaces the most similar resources in the list with maximally dissimilar ones. It was found that although this reduced performance in terms of predictive accuracy, overall user satisfaction had increased (at specific levels of diversification).

Authority: The authority of a resource is a measure of its centrality and importance, relative to all other resources in a dataset. Authority can be defined internally, or imposed externally. In this work, the author chose to view authority as an ordering imposed on a domain by an external judging entity – ISI's Impact Factor [37].

Personalisation: Personalisation measures assess the overlap between recommendation lists provided for each user. Spearman's Footrule for Top–$k$ lists is a mathematically rigorous metric, used to perform the comparison [25]. For two lists, $\sigma_1$ and $\sigma_2$, the Footrule, $F(\sigma_1, \sigma_2)$ is given by:

$$F^l(\sigma_1, \sigma_2) = \sum_{i \in \sigma_1 \cup \sigma_2} |rank(i, \sigma_1) - rank(i, \sigma_2)|, \tag{2.6}$$

where $rank(i, \sigma_j)$ is the rank of resource $i$ in list $\sigma_j$. For $i \notin \sigma_j$, $rank(i, \sigma_j)$ returns a constant, $l$, set to $k+1$ in this work. The personalisation score for the algorithm is the mean $F^l$ over all pair comparisons.

Rank, Coverage, and Adjusted Rank: In a leave–1–out methodology, Rank is defined as the ranking of the withheld resource in the recommendation list produced. To aggregate results the percentage of the time that a withheld item appears within a given threshold is measured. It is similar to the Precision metric (discussed in Section 2.2.1) but is not concerned with the value of the predicted rating but rather its position in the list. Coverage expresses how many times the withheld resource appears in the recommendation list out of all the tests made. While related to Recall (also discussed in Section 2.2.1), Coverage does not require the total number of positive search results to be known at the outset. Finally, Adjusted Rank is the product of Rank and Coverage giving an effective rank measure which penalizes a RS for the times the withheld item does not appear in the list of recommendations.

Boldness: How strongly an algorithm recommends a particular resource to users is measured by boldness. It compares how frequently the system predicts a resource will receive ratings at the extremes of the rating scale to how frequently such extreme predictions are expected.

$$Boldness = \frac{\text{\% of actual extreme predictions}}{\text{\% of expected extreme predictions}} \tag{2.7}$$

A flat probability distribution is suggested as a model for the expected frequency of extreme ratings, but statistics from the data could also be used, along with other probability distributions.

Adaptability: This metric is designed to capture how modifications made to a user's profile are reflected in the recommendation lists provided to them. To measure this, users are grouped in pairs, and their profiles split into four segments of equal size. For each pair of users, three pseudo–users are generated: Adapt–Light, Even Split, and Adapt–Heavy. Adapt–Light pseudo–users inherit three segments from the first user and one from the other, Even Split ones have profiles that consist of two segments from each user, while Adapt–Heavy pseudo–users are constructed using one segment from the first user and three from the other. The adaptability score is obtained by comparing the distance between recommendation lists for the pseudo–users and the last (unused) segment of the second user's profile to the distance between the profile of the first user and the same segment. Spearman's Footrule is used as the distance metric.

In order to use HRI as a constructive theory, the desired qualities of the system under development must first be mapped to specific aspects. The metrics associated with these aspects by HRI can then be used to benchmark and evaluate the recommendation algorithms under consideration.

## 2.3 Web 2.0 and Social Computing

In [39], Hogg et al. state that Web 2.0 is "*the philosophy of mutually maximizing collective intelligence and added value for each participant by formalized and dynamic information sharing and creation*". In short, Web 2.0 promotes the concept that users can act simultaneously as both the creators and the consumers of information, if provided with the platforms and tools to facilitate collaboration. Developments such as social networking sites [48, 94, 93], wikis [87], blogs [33, 86], Web APIs [61], and Web Applications [34, 35] all fall under this category.

Web 2.0 is not associated with any particular technological advances, but rather with the realisation that people are willing to publish personal information freely, and *en masse* on the Web. This enthusiastic behaviour is probably associated with a number of factors such as the desire to connect and share experiences with similarly minded people, or expectations of extracting added value in terms of information organisation and retrieval capabilities offered by the publishing systems (and powered by their collaborative nature). Since the unwillingness of users to provide ratings and personal information is seen as one of the factors hindering the performance of current implementations, Recommender Systems have much to gain by exploiting and integrating information their users publish using technologies associated with Web 2.0.

## 2.4 The Semantic Logger

The Semantic Logger (SL) is an autobiographical metadata acquisition system, and the outcome of previous work, [89]. The intuition behind the system is to capture any information created, or accessed by its' users on their local machine, or online. The information captured is then integrated to assemble the context associated with each particular event. Such associations are inferred when objects occur at the same time or place, and the assembled context can be used to produce annotations. It can be seen as a means to populate the Semantic Web [9] with personal metadata that is automatically generated for raw multimedia data captured by personal devices. Since analysing multimedia objects to obtain high-level descriptors is a hard task, it may be easier to first associate a multimedia object with other objects for which semantics can be more readily extracted. This aspect of the work is described in [23]. However it has been developed with the user profiling component of the framework proposed in this dissertation in mind.

The Semantic Squirrel Special Interest Group (SSSIG) [40] is a group of researchers based at the University of Southampton who aim to automate the process of logging available raw data, (or *'nuts'*), that can describe aspects of ones personal experience. A number of squirrels have been developed in this process, and an ethos of the group is to preserve this raw data in order to retain any unforeseen potentials for exploitation and transcend issues pertaining to platform and application restrictions. The SSSIG is also focusing on identifying novel systems using the collected data.

This raw data forms the basis of the knowledge acquisition phase for the Semantic Logger and is parsed into RDF[47] representations. Effort has been put in selecting appropriate representations: they have been taken from proposed standards at the W3C or other standards making bodies, or have been selected due to current uptake on the web. Where such standards have not been available, local ontologies were constructed which describe the given phenomenon, while maintaining simplicity and generality. The intent is to use raw data about people in order to build the context of a particular event at a particular time. By virtue of the fact that each event logged by the system is time-stamped, different levels of granularity can be chosen to describe its context.

The majority of the information held at a private machine is expected to be in textual form, such as e–mails, calendar entries, articles, web–pages, etc. However, this information is of limited use if it is kept in its original form, as contiguous sentences. We use techniques from the field of Natural Language Processing to deal with the issue of extracting semantics from such fragments, by performing tasks such as *named entity recognition* and *part–of–speech tagging*. GATE [17], the General Architecture for Text Engineering, is comprised of an architecture, a free open source framework and a graphical development environment to support such tasks. In the context of this work, GATE is used to identify concepts and named entities in text, that can then be used as profiling features.

### 2.4.1 Sources of Information

A distinction is made between *personal metadata* – information captured and stored directly by the user, and information harvested from the Internet – *social metadata*. Figure 2.2 presents the sources of information exploited by the system, grouped under these categories.

#### 2.4.1.1 Personal metadata

The items grouped under the label, *Personal Context*, are sources of contextual information logged directly by the user.

- *Web Browsing:* Information regarding the web-browsing, bookmarking, and downloading activities is captured. These are parsed into a local-namespace which utilises concepts described in the Netscape ontology[1].
- *Location Information:* GPS tracklogs are converted into geographic points, expressed in the *Basic Geo (WGS84 lat/long) Vocabulary.* Location information is also tracked by logging wireless access points a user's laptop connects to. This is done using the online service Plazes[68].
- *Calendar entries:* We have adopted the W3C recommendation in representing calendar entries in RDF. A client-side application is available for download from

[1]http://www.mozilla.org/rdf/doc/

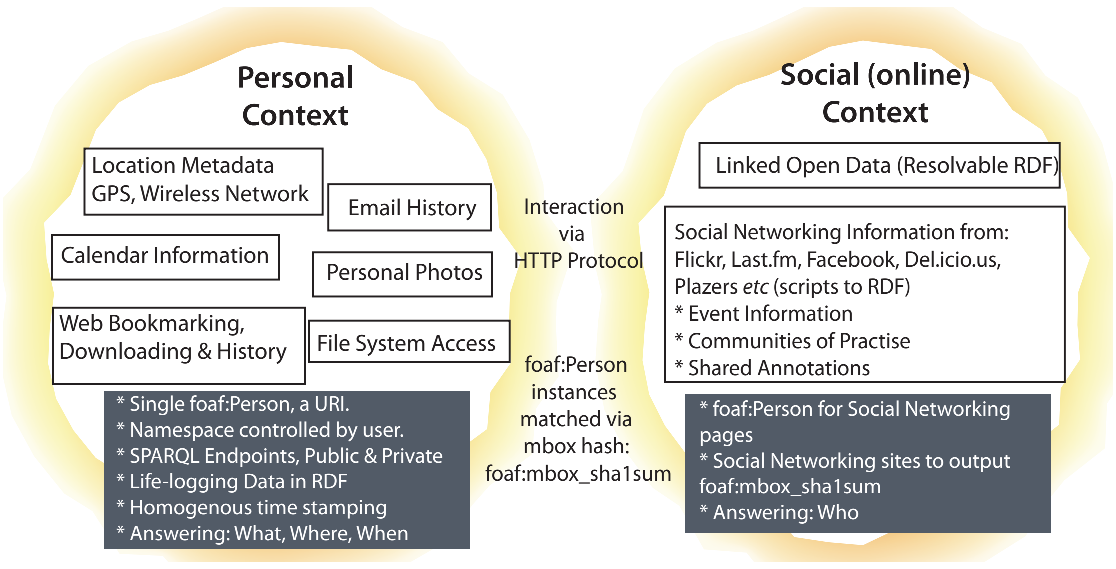

Personal Context
Location Metadata GPS, Wireless Network
Email History
Calendar Information
Personal Photos
Web Bookmarking, Downloading & History
File System Access
* Single foaf:Person, a URI.
* Namespace controlled by user.
* SPARQL Endpoints, Public & Private
* Life-logging Data in RDF
* Homogenous time stamping
* Answering: What, Where, When
Interaction via HTTP Protocol
foaf:Person instances matched via mbox hash: foaf:mbox_sha1sum
Social (online) Context
Linked Open Data (Resolvable RDF)
Social Networking Information from: Flickr, Last.fm, Facebook, Del.icio.us, Plazers etc (scripts to RDF)
* Event Information
* Communities of Practise
* Shared Annotations
* foaf:Person for Social Networking pages
* Social Networking sites to output foaf:mbox_sha1sum
* Answering: Who

FIGURE 2.2: Overview of the Semantic Logger Architecture

the Semantic Logger site to automate the export of iCal [18] files (commonly used and platform independent) into this representation. Calendar entries can serve as context indicators for user activities, or geographical locations.

- *Email History:* The user's MBOX is parsed into a local namespace, detailing e–mail activity. These can in turn be matched up to public `foaf:mbox_sha1sums` to indicate relationships between people.

These are parsed into RDF, associated to a user's URI, and stored in a personal knowledge–base.

#### 2.4.1.2 Social metadata

Information harvested from the internet, and uploaded to the SL, is what we refer to as *social metadata*. The sources of information selected to be captured from Web have been chosen to complement the sources of *personal metadata* described earlier by utilising the social aspect of the sites. The information gathered by the system includes but is not limited to:

- *Web Browsing Information:* The site del.icio.us, [93], provides information about Web bookmarking. This information is parsed into the Netscape ontology.

- *Location Information:* Location information is also taken from the photo-sharing site Flickr, in the form of geo-tagged images.

- *Music Playlist Information:* Last.fm is used to capture information regarding the user's music listening behaviour.

As it stands the Semantic Logger consists of a central public knowledge-base where all users publish their public data, along with two knowledge-bases for each user that are both password protected. One is set to be the user's private KB while the second one can also be accessed by the user's friends.

## 2.5 Machine Learning algorithms

This section presents various Machine Learning algorithms that are typically used to drive the process of learning user profiles and producing recommendations. The selection of algorithms presented here is far from an exhaustive listing, since the vast majority of such algorithms can be modified for a recommendation setting, but rather serves the purpose of illustrating the spectrum of applicable techniques and their intuitive differences.

### 2.5.1 Deterministic algorithms

Deterministic machine learning algorithms make use of the Vector–Space model. The model was first expressed in Information Retrieval (IR), in order to map *queries* to *documents*. Both concepts are seen to consist of *terms*, the words that make up a document or a query. In the Vector–Space model, documents are represented as vectors in a multidimensional Euclidean space. Each axis in this space corresponds to a term, and the coordinate of a document in the direction of a term is determined by the number of times the term occurs in the document.

This approach can be applied to Recommender Systems in various ways, depending on the recommendation strategy used. For CB systems, each item is seen as a vector of descriptive features (e.g. colour). Specific attributes (e.g. black) can then be mapped to numerical values in order to obtain co–ordinates in the direction of the feature. Users can then be represented as the centroid of the points representing items in their profiles.

In CF, users are seen as vectors with dimensionality equal to the number of resources available for recommendation. The ratings provided by a user then give the coordinates in the directions of the rated resources. The user for whom recommendations are to be made is treated as a query to retrieve similar users. A similar process is carried out in DF, using vectors of demographic features to represent users.

The following sections elaborate on specific algorithms that make use of this conceptualisation.

#### 2.5.1.1 $k$–nearest neighbours and $k$–means clustering

$k$–nearest neighbours ($k$–NN) is the most popular algorithm for implementing Collaborative Filtering, as it is very intuitive to implement. Once users have been represented as vectors in a multi–dimensional space, distance between them can be evaluated using a variety of metrics, such as the Euclidean distance or vector cosine similarity [71]. The distance between each pair of users is then calculated, and each user associated with a set of $k$ most similar users; this is called the user's *neighbourhood.* Ratings provided by the neighbourhood are then aggregated, and used to produce recommendations.

A similar intuition is expressed by the $k$–means clustering algorithm. The objective is to produce clusters of similar users by assessing the distance between their representative vectors, similarly to $k$–NN. However, the target number of clusters is defined at the outset, instead of the neighbourhood size. It is an iterative algorithm, where each cluster is associated with its centroid, which is initially an arbitrary vector. Each user is associated to the cluster which minimises the distance between the user and its centroid. Once all users have been assigned to clusters, the centroid for each cluster is recomputed, and the algorithm proceeds to the next iteration. Once the assignment of users to clusters ceases to change (by much), the algorithm terminates. This approach provides computational benefits over $k$–NN, provided that the number of clusters remains relatively small compared to the number of users.

#### 2.5.1.2 Singular Value Decomposition and Latent Semantic Indexing

In the Vector–space model, each user was considered a vector, in a multidimensional space where each resource was allocated a distinct dimension. The rating provided by user about a resource then provided the vector's coordinate at that particular dimension. This description appears redundant: the users will likely exhibit similar rating behaviour across sets of resources. For example fans of a particular music group could rate all their releases highly. Singular Value Decomposition (SVD) [31] is a technique used in order to eliminate redundancy and reduce the dimensionality of the representation. For an $m \times n$ matrix $A$, SVD is a standard decomposition of the form:

$$A_{m\times n} = U_{m\times r} \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{pmatrix} V^T_{r\times n} \tag{2.8}$$

where $r \leq n$, and the matrices $U$ and $V$ are column ortho–normal, $U^TU = V^TV = \mathbf{I}$. $\Sigma$, the diagonal $m \times r$ matrix in the middle, is the matrix of singular values $\sigma$, which is typically organised such that $\sigma_1 \geq \cdots \geq \sigma_r \geq 0$, by making the corresponding alterations to $U$ and $V$. Now, each row of $U$ can be considered an $r$–dimensional representation of the corresponding row of $A$ and similarly each column of $V^T$ an $r$–dimensional representation of the corresponding column of $A$. Thus, calculating similarity between users can be carried out by computing the similarity between the corresponding rows of $U$.

In Latent Semantic Indexing (LSI), [19], the corpus is first used to compute the matrices $U$, $\Sigma$, and $V$. Then, a number of dimensions corresponding to the smallest singular values are dropped, and similarity between users is computed for collaborative filtering using the reduced representations. An interesting feature of this approach is that both users and items are projected onto the same space. Therefore a single comparison can give the similarity between an item and a user profile.

### 2.5.2 Probabilistic algorithms

Although the vector–space model and related technologies are very well established within the Information Retrieval community, it remains an *operational* model. It provides a precise definition for computing similarity between documents, but makes no attempt at justifying *why* relevance should be defined in this way. The following sections present probabilistic approaches, based on statistical models for the generation of documents and terms; in the RS case, users and resources respectively.

#### 2.5.2.1 Bayesian methods

The application of Bayesian methods in classification has had a large impact on the field of Machine Learning, and numerous studies have been conducted to assess its optimality, e.g. [22]. Such methods are driven by Bayes' rule of conditional probabilities:

$$P(r|u_1, \ldots, u_n) = \frac{P(r)P(u_1, \ldots, u_n|r)}{P(u_1, \ldots, u_n)} \tag{2.9}$$

Where $r$ is a *type* (or *class*), and $\{u_1, \ldots, u_n\}$ *tokens* (or *instances*). Bayes' rule calculates the probability that tokens in $\{u_1, \ldots, u_n\}$ have been generated by $r$, based on prior probability distributions over types and tokens, and a generative model for each type to provide the likelihood.

Conditional probabilities provide a natural way to address issues pertaining to causality, as the question of similarity between tokens becomes the probability they belong to the same class.

By assuming that tokens have been generated independently, i.e.:

$$P(u_1, \ldots, u_n) = \prod_{1 \leq i \leq n} P(u_i) \tag{2.10}$$

and also for each class $r$:

$$P(u_1, \ldots, u_n|r) = \prod_{1 \leq i \leq n} P(u_i|r) \tag{2.11}$$

equation 2.9 becomes:

$$P(r|u_1, \ldots, u_n) = \frac{P(r)P(u_1|r) \ldots P(u_n|r)}{P(u_1) \ldots P(u_n)} \tag{2.12}$$

Because this assumption is unlikely to hold in most situations, the method has been dubbed the 'naive' Bayes classifier. Despite its naivety the algorithm has frequently been found to outperform its deterministic counterparts.

The 'naive' Bayes classifier can be intuitively adapted to produce recommendations. Let the tokens $\{u_1, \ldots, u_n\}$ be the resources rated by the active user, and $r$ the resource being assessed for recommendation. $P(r)$ and $P(u_1) \ldots P(u_n)$ are considered to be the popularity of each resource; this is defined as the ratio of users that rated them:

$$P(r) = \frac{\text{number of users who rated resource } r}{\text{total number of users in the system}} \tag{2.13}$$

Finally, the probabilities $P(u_1|r) \ldots P(u_n|r)$ can be defined as the number of times resources $u_i$ and $r$ have been rated together:

$$P(u_i|r) = \frac{\text{number of users who rated resources } u_i \text{ and } r}{\text{total number of users in the system}} \tag{2.14}$$

However, in the RS setting the independence assumption is analogous to the statement that the ratings provided by users are independent to each other. In other words, the resources that a user chooses to rate are a set drawn randomly from the corpus, which is clearly untrue. Chapter 5 details a method to encode such dependencies in a graph, which is subsequently used to obtain both the prior distributions and the likelihood.

#### 2.5.2.2 Expectation – Maximisation and Probabilistic Latent Semantic Indexing

Probabilistic Latent Semantic Indexing, [38] is a probabilistic extension to standard Latent Semantic Analysis and relies on decomposing the dataset using an aspect model rather than applying SVD. The probabilistic nature of PLSI provides a solid statistical background and defines a generative data model.

An aspect model is a statistical latent variable model [77] to associate an unobserved class, to each observation pair. In terms of a joint probability model it can be expressed as :

$$P(d, w) = P(d)P(w|d) \tag{2.15}$$

$$P(w|d) = \sum_{z \in Z} P(w|z)P(z|d) \tag{2.16}$$

where $z$ represents the latent classes, while $d$ and $w$ encode for the observations in terms of documents and words respectively. To generate an observed pair $(d, w)$ under this model, the document is selected first with probability $P(d)$. Then, the word $w$ is generated with probability $P(w|d)$. This is the sum of the probabilities that the word is generated by each latent class $z$, scaled by the probability that the document was also generated by the same class [54]. In the RS context, documents are identified with users, while words are represented by the resources assessed for recommendation.

In short, the intuition expressed by the algorithm is the following. Both users and resources are generated by the same set of classes. To calculate the probability of observing a particular rating (a $(user, resource)$ pair), the probabilities of each class having generated both the user and the resource need to be obtained. Their product gives the probability that a particular class has generated both events. The sum of these products will then give the probability of the resource being generated by the same latent class as the user.

Two independence assumptions are made here:

- Observations $(d, w)$ are generated independently. This is equivalent to the *'document–as–bag–of–words'* approach commonly used in information retrieval.

$$P(d, w|d', w') = P(d, w) \tag{2.17}$$

- Conditional independence is assumed, stating that given the latent class $z$ the observed variables $d$ and $w$ are independent.

$$P(d_1, \ldots, d_n|z) = \prod_{1 \le i \le n} P(d_i|r) \tag{2.18}$$

$$P(w_1, \ldots, w_n|z) = \prod_{1 \le i \le n} P(w_i|r) \tag{2.19}$$

The estimated values of $P(d)$, $P(z|d)$ and $P(w|d)$ are approximated by following the likelihood principle and maximising the log-likelihood function:

$$L = \sum_{d \in D} \sum_{w \in W} n(d, w) \log(P(d, w)) \tag{2.20}$$

where $n(d, w)$ is the number of occurrences of word $w$ in document $d$.

Expectation Maximisation (EM) is the standard procedure with which the maximum likelihood is estimated [20] for the complete data, including the unobserved variables. This is a two-step iterative process where the posterior probabilities, $P(z|d, w)$, for the latent classes, $z$, are computed in the expectation step. The model parameters are then re–estimated for the posteriors previously computed and available statistics from the training set in the Maximisation step.

Assuming that observations $(d, w)$ are conditionally independent, the Expectation (E) step is a consistent definition of $P(z|d, w)$. Given the posterior probabilities for all latent classes and the term frequency vectors for all documents, the Maximisation (M) step is defined by finding the maximum of the Likelihood function. This is achieved by partially differentiating 2.20 with respect to each variable and equating the result to 0.

The computational requirements of this process can be drastically reduced through the use of summarisation techniques, such as mrkd–trees, as shown in [60, 91].

### 2.5.3 Graph–based algorithms

Graph–based algorithms made their way into IR systems as the size of the Web and the reach of search engines increased rapidly, and traditional systems applied to Web data were starting to perform badly. The reason for this shortcoming is the combination of a corpus growing at an immense rate, while the queries submitted contained only a small number of words. The difficulty was in returning a handful of the most relevant and credible documents, when millions could potentially match the query. Graph–based algorithms express a different notion of similarity: Documents are not only related to each other by the fact the use the same terms, but also because they refer to each other using hyper–links. The following sections present two of the most influential such algorithms.

#### 2.5.3.1 Hubs, Authorities, and HITS

Inspired by bibliometry, and the notions of *prestige* and *authority*, Kleinberg [44] proposed a similar conceptualisation for the Web. The web was represented as a directed graph with nodes encoding Web pages, and edges hyper–links. It was postulated that the Web includes two types of popular pages: *authorities*, which carry definitive and high quality information, and *hubs*, comprehensive lists of authorities. Further, each page was considered to be both a hub and an authority, to some extent. In order to calculate the significance of each page, both as a hub and an authority he introduced an iterative algorithm, Hyperlink Induced Topic Search (HITS).

HITS makes use of a standard IR system to extract a subgraph from the Web relevant to a query. Each node is initially assigned unit authority and hub scores. Then, the algorithm iterates over each node in the graph, carrying out the following operations:

1. The authority score of each node is the sum of the hub scores of the nodes linking to it.
2. The hub score of each node is the sum of the authority scores of the nodes it links to.

The iteration continues until there is no significant change in the scores associated with each node. It can be shown by the spectral decomposition theorem [36] that if $G$ is the adjacency matrix encoding the graph, the principal eigenvector of $G^TG$ will contain the authority scores for each node. Respectively, the principal eigenvector of $GG^T$ will contain the hub scores for each node.

#### 2.5.3.2 PageRank

PageRank, [13], expresses a similar intuition to HITS. Every page on the Web is considered to have a measure of *prestige*. The algorithm emulates a Web surfer, clicking on hyper–links at random, forever. The surfer starts from a random node, and clicks on one of the links on that page at random, with equal probability. Once the surfer has followed infinite links, the frequency at which each page was visited is proportional to its prestige.

If the graph is *irreducible* and *aperiodic*, the principal eigenvector of the adjacency matrix would provide the prestige value of each page. In other words there would have to be no nodes in the graph without out–links, and no cycles in which the surfer could get trapped in. Since the web is not strongly connected and aperiodic, the situation is rectified by applying a random surfer model, where the surfer may jump between any two nodes, with a small probability $d$, in addition to following existing links with probability $1 - d$. If the surfer reaches a node with no out–links, a jump is made to another node at random, with probability 1.

PageRank is used by Google, to rank the results obtained in response to a query from its text indices of web pages. Since it is independent of the content of a page, or the query, PageRank is pre–computed for all pages that Google indexes. When a query is received, the text index is used to obtain a set of possible results. Then, the results are ranked using an undisclosed combination of each page's PageRank score with the textual match score.

## 2.6 Analysis and conclusions

Although research on Recommender Systems has been an active field of research for a number of years, and various commercial systems have been deployed, such systems suffer the effects of well known problems associated with RS technologies. These include data sparsity, cold–start problems, and insensitivity to context among others.

The strong roots of the field in Machine Learning contributed neglecting the social nature of recommending, as exemplified by the content–based, and rule filtering approaches. Collaborative and demographic filtering techniques were proposed to remedy this situation, while the development of context aware systems provides opportunities for situated recommendations. However, the focus on e–commerce applications and the use of RS technologies as a tool to maximise vendor profits leads to new considerations, such as potential biasing effects and a narrow spread of recommendations. Such issues are addressed in detail by the Human Recommender Interaction Theory, [55].

The limited amount of literature available on performing cross–domain recommendations points out that information from multiple domains can be leveraged to alleviate some of the problems presented in Section 2.1.8.

The emergence of the Web 2.0 phenomenon provides immense opportunities for RS to collect information about their users; especially so since the reluctance of users to provide profiling information, or even ratings is well documented, [79]. At the same time Semantic Web technologies provide well defined methods and tools for the seamless integration of disparate data. These can be exploited to integrate profiling information about users, potentially harvested from Web 2.0 and social networking sites.

Finally, since graph–based algorithms have been empirically shown to be extremely efficient in ranking vast collections of documents on the Web, their application to Recommender Systems could provide fascinating enhancements. Although hyper–links are not present, one could assume edges between users and rated resources, and between resources and descriptive features, in order to apply such algorithms in a RS setting.

# Chapter 3

# Methodology

As pointed out in Section 1.1 there are benefits to be reaped by enabling Recommender Systems to consider multiple domains to draw recommendations from. First, the assumption that the purpose of deploying such systems is to maximise product sales is dropped. Instead, information made freely available through the wide–spread adoption of technologies associated with Web 2.0 and social networking sites is imported into the system to replace the notions of a product catalogue and subscribed users. Since graph–based algorithms have been highly successful in ranking documents on the Web based on their authority, a similar algorithm is developed and applied to the task of making recommendations. Moreover, the profiles used to represent users for which recommendations are to be made are automatically compiled through the unobtrusive monitoring of user activities, both *online* and *offline*.

This chapter details the underlying methodology of the recommendation approach presented herein. The terminology used in the remainder of this dissertation is provided first, and the terms established are used to present the recommendation process carried out by the framework. The methods used to assess the various elements are then provided, establishing an evaluation methodology. Finally, the expected characteristics of the framework are contrasted with the shortcomings exhibited by RS technologies that were identified in the previous chapter.

## 3.1 Terminology

**Definition 1. Resource**
A piece of information whose utility can be evaluated subjectively by a human agent. We denote the set of all resources $R$, and its size $|R|$.

**Example 1.** The film 'Gone with the wind', the music of 'Led Zeppelin', or the 'The Semantic Web' article by Berners Lee et. al. can all be considered resources.

**Definition 2. User profile**
User profiles are sets of resources that a user of the system has previously expressed interest in. $U$ represents the set of all users, and the function $\rho$ maps a user to the resources in his profile.

$$\begin{aligned} \rho \ : \ U &\rightarrow 2^R \\ u &\mapsto \rho(u) = \{r_1, r_2, \cdots, r_n\}, \quad 1 \leq n \leq |R| \end{aligned} \tag{3.1}$$

$2^R$ denotes the *powerset* – the set of all subsets – of $R$.

**Example 2.** A person, $u$ who is using the system and has downloaded the 'The Semantic Web' article, has accessed the IMDb[41] page for 'Gone with the wind', and has a folder on their hard drive called 'Led Zeppelin' may have the profile:

$$\rho(u) = \{\text{'The Semantic Web', 'Gone With The Wind', 'Led Zeppelin'}\}$$

**Definition 3. Experts**
The term expert refers to a human agent that explicitly expresses preferences regarding resources in a subset of $R$. The function $\rho'$ maps an expert to the resources he has assessed. In addition, the function $\nu$ can be applied to an expert–resource pair, $(e, r)$ to obtain a numerical value, indicative of the experts' relative preference over the assessed resource $r$. Each such mapping is referred to as a *rating*. The set of all experts is denoted $E$.

$$\begin{aligned} \rho' \ : \ E &\rightarrow 2^R \\ e &\mapsto \rho'(e) = \{r_1, r_2, \cdots, r_n\}, \quad 1 \leq n \leq |R| \end{aligned} \tag{3.2}$$

$$\begin{aligned} \nu \ : \ E \times R &\rightarrow \mathbb{R} \\ e, r &\mapsto \nu(e, r) = w \end{aligned} \tag{3.3}$$

Having introduced experts, the function $\varepsilon$ can now be defined to provide a mapping from a resource to the experts that rated it.

$$\begin{aligned} \varepsilon \ : \ R &\rightarrow 2^E \\ r &\mapsto \varepsilon(r) = \{e_1, e_2, \cdots, e_n\}, \quad 1 \leq n \leq |E| \end{aligned} \tag{3.4}$$

As such, experts in $\varepsilon(r)$ will have provided ratings for resource $r$:

$$\forall e \in \varepsilon(r), \quad \nu(e, r) > 0$$

**Example 3.** To illustrate this concept we will refer to two online services we have incorporated in the framework: Last.fm [48] and NetFlix [6], from the domains of music and film, respectively. Last.fm monitors the music its users listen to, and records how many times each track, album or artist has been played by each user. NetFlix is an online DVD rental company which allows users to rate films they have seen on a scale from 1 to 5. They have now opened a large portion of their dataset to the public and announced a competition to improve their recommendation algorithm [63]. In the case of Last.fm, the function $\rho'(e)$ can be used to obtain the list of all artists, albums, and tracks that a particular Last.fm subscriber, $e$, has listened to, while $\nu(e, r_i)$ will return the number of times $e$ has played item $r_i$. Conversely, for NetFlix subscribers the functions $\rho'$ and $\nu$ return the movies rated and the rating associated with each {expert, movie} pair, respectively.

We can now clarify the use of the term *domain* in this context.

**Definition 4. Domain**
Domains are openly accessible corpora of ratings. As such, each domain is characterised by a group of experts who express preferences on a subset of the resources in $R$. The function $\rho^*$ maps a domain to the resources for which preferences are expressed upon, while $\varepsilon^*$ returns the set of experts that have evaluated them. It is important to note that resources (and experts) may belong to more than one domain. $D$ represents the set of all domains.

$$\begin{aligned} \rho^* \quad : \quad D & \quad \rightarrow \quad 2^R \\ d & \quad \mapsto \quad \rho^*(d) = \{r_1, r_2, \cdots, r_n\}, \quad 1 \leq n \leq |R| \end{aligned} \tag{3.5}$$

$$\begin{aligned} \varepsilon^* \quad : \quad D & \quad \rightarrow \quad 2^E \\ d & \quad \mapsto \quad \varepsilon^*(d) = \{e_1, e_2, \cdots, e_m\}, \quad 1 \leq m \leq |E| \end{aligned} \tag{3.6}$$

$$. \quad \forall e \in \varepsilon^*(d), \quad \rho^*(d) \cap \rho'(e) \neq \emptyset$$

**Example 4.** Last.fm and NetFlix are good domain examples. In both cases $\varepsilon^*$ returns the subscribers of these two services. $\rho^*$ resolves all artists, tracks, and albums available for Last.fm, while for NetFlix it returns the company's DVD catalogue. Furthermore, 'Led Zeppelin' can be expected to appear in both domains, as various documentaries and music videos of the band are available. A less obvious example domain is CiteSeer[27], in the academic publication domain. Here, authors can be considered to be both the experts and the resources. The number of times one author references another can then be used instead of ratings or playcounts.

We can now formulate the objective: **To develop a framework able to predict the utility of any *resource*, irrespective of its *domain*, to a particular *user*.** This formulation introduces the requirement that a user profile is to be compared with those of *experts* in various *domains*. The difficulty arises when the resources to be evaluated occur in a different domain to those which contain resources in the users' profile, since it is not clear how the user should be projected in the new space. More formally, this situation occurs for domain $d$ and user $u$ when:

$$\rho(u) \cap \bigcup_{e \in \varepsilon^*(d)} \rho'(e) = \emptyset \tag{3.7}$$

This obstacle indicates the need for a homogeneous vocabulary of terms, which can be used to identify any resource in the world. Provided this is available, projections of both the user and experts in $\varepsilon^*(d)$ on this vocabulary can be obtained and compared. Such a vocabulary is termed a *Universal Vocabulary*. The intuitive absurdity of such a requirement is addressed in Chapter 4.

**Definition 5. Universal Vocabulary and associated metrics**
A vocabulary that allows the unique identification of any resource in the world through a combination of an arbitrary number of terms contained within it. Furthermore, such a corpus encodes the semantic relations between terms in order to effectively compute similarity between (the projections of) users and resources, independently of their domain of origin. In order to express this, we denote the set of terms contained in the Universal Vocabulary $W$, and introduce a further three functions: $\varphi$, $\varphi^*$, and $\delta$. The function $\varphi$ projects a resource onto the Universal Vocabulary, returning the set of terms that can be used to describe it, while $\varphi^*$ operates on a set of resources to obtain the smallest subset of the vocabulary that is sufficient to describe all resources in the former set. $\delta$ is a similarity metric, which assesses the strength of the relationship between a particular term and a collection of other terms. This is required since we intend to compare a user's profile (which is seen as set of resources) to each resource being assessed for recommendation.

$$\begin{aligned} \varphi \quad : \quad & R && \rightarrow && 2^W \\ & r && \mapsto && \varphi(r) = \{f_1, f_2, \cdots, f_n\}, \quad 1 \leq n \leq |W| \end{aligned} \tag{3.8}$$

$$\begin{aligned} \varphi^* \quad : \quad & 2^R && \rightarrow && 2^W \\ & \{r_1, r_2, \cdots, r_n\} && \mapsto && \varphi^*(\{r_1, r_2, \cdots, r_n\}) = \{f_1, f_2, \cdots, f_m\} \\ & && && , \quad 1 \leq n \leq |R|, \quad 1 \leq m \leq |W| \end{aligned} \tag{3.9}$$

$$\begin{aligned} \delta \quad : \quad & W \times 2^W && \rightarrow && \mathbb{R} \\ & \{f, \{f_1, f_2, \cdots, f_n\}\} && \mapsto && \delta(f, \{f_1, f_2, \cdots, f_n\}) = s, \quad 1 \leq n \leq |W| \end{aligned} \tag{3.10}$$

**Example 5.** A graph–like structure with vertices representing terms in the vocabulary and edges encoding for the semantic relationships between terms, such that each resource can be mapped to a unique subgraph. Such structures are attractive for our purpose since they can be exploited to provide intuitive definitions for $\delta$, the similarity metric.

As presented in detail in Chapter 4, Wikipedia is used as the universal vocabulary for the purposes of this dissertation. Articles in Wikipedia are used as the terms in the vocabulary, while the presence of a hyperlink between two articles is assumed to encode an implicit semantic relationship between the concepts they describe. Thus, the function $\varphi(r)$ encodes a process by which resource $r$ is mapped to one (or more) Wikipedia articles $f_i$ that describe it. Similarly, $\varphi^*$, operates on a set of resources and returns a corresponding set of Wikipedia articles. A simplistic definition of $\delta$ can for example be the average number of hyperlinks that would have to be followed from articles in $\{f_1, f_2, \ldots, f_n\}$ to reach article $f$. Chapter 5 provides a probabilistic model for assessing $\delta$ based on the traversal of hyperlinks between Wikipedia articles to assess similarity between them.

## 3.2 The recommendation process

The recommendation process can be seen as consisting of four, non–sequential stages, as shown in Figure 3.1. First, a complete (as much as possible) user profile is required. In addition, the user's contextual setting is evaluated at run–time to identify the domain from which to draw resources for recommendation. Section 5.4.1.4 provides the specifics of this process. Then, the preferences of an external community (or social network) pertaining to resources in the domain, are obtained and processed. The process of using Wikipedia as a mediator between expert preferences and elements of user profiles from multiple domains is detailed in Chapter 4. Ratings provided by the domain experts, and hyper–links in Wikipedia, are integrated as edges in a graph. A Markov chain [50] is then used to produce recommendations by assessing the probability of traversing the graph towards a particular resource node, using the nodes in the users profile as a starting point. This model is the machine learning component of the framework, and Chapter 5 presents details on its construction and usage.

## 3.3 Rationale and methods for evaluation

The methodology presented in this chapter consists of various elements, that are amalgamated by the framework to produce recommendations. This section provides the rationale for the evaluation of each such element, and outlines the methods used to carry it out.

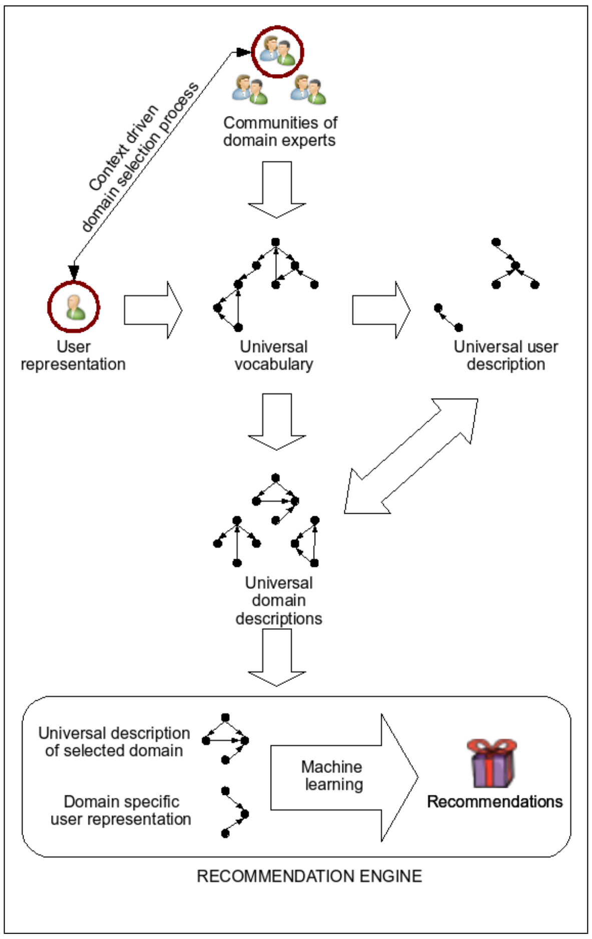

Context driven domain selection process
Communities of domain experts
User representation
Universal vocabulary
Universal user description
Universal domain descriptions
Universal description of selected domain
Domain specific user representation
Machine learning
Recommendations
RECOMMENDATION ENGINE

FIGURE 3.1: Overview of the recommendation process.

### 3.3.1 Universal vocabulary

When assessing a corpus to be used as a universal vocabulary it is of paramount importance to ensure that similarity between domain resources is preserved when they are projected to this new space. It is relatively simple to define a procedure to assess this. Initial similarity values between domain resources can be obtained by using the standard vector space approach, as used in item–based CF. Then a new vector is constructed for each resource, encoding connections between the resource and other terms in the vocabulary. Similarity values between these new vectors can then be compared to the initial values. The strength of the correlation is then used as evidence of whether or not similarity has been preserved.

The assumption that each resource available for recommendation can be associated with a single term in the universal vocabulary is a large one to make, as only resources that can be projected onto this vocabulary can be recommended. In particular, since Wikipedia is proposed as the universal vocabulary in our approach, it would be assumed that a Wikipedia article describing the resource exists. However, taking into account that resources are imported to the framework from external social networks and communities, together with the fact that most such communities allow their members to associate 'tags' to resources, provides an opportunity to relax the assumption. Provided that textual descriptors in the form of tags are available, the set of Wikipedia articles that correspond to them can be used as the universal description of the resource. The effectiveness of this process needs to be assessed.

These issues are addressed in Section 4.3

### 3.3.2 Cross–domain recommendations

In broad terms, by cross–domain recommendations we refer to the act of recommending resources from a domain different from those to which the features that appear in a user profile belong. To illustrate the process, consider two domains, $d$ and $d'$. Also let $\rho(u)$ be a subset of $\rho^*(d)$, and $rec_{d'}(\rho(u)) \subset \rho^*(d')$ denote the recommendations from domain $d'$ produced by the framework based on the profile $\rho(u)$. The framework can then be applied again to obtain $rec_d(rec_{d'}(\rho(u))) \subset \rho^*(d)$; recommendations from $d$ based on the ones obtained in the previous step. The magnitude of the overlap between the original profile, $\rho(u)$, and the final recommendation list $rec_d(rec_{d'}(\rho(u)))$ can serve as evidence for the consistency of the framework. Performance is expected to degrade as more 'intermediary' domains are added to the process described above. The rate at which performance is lost is indicative of how well the information contained in the original profile is retained by the framework while producing cross–domain recommendations. The application of this experimental approach is provided in Section 5.3.2.

### 3.3.3 User profiling

User profiling in the framework is carried out through the unobtrusive monitoring and archival of information accessed, or created by the users. As such, there is a need to demonstrate taht the profiles generated are adequate for use in producing recommendations.

To carry this out, the set of experts $\varepsilon^*(d_i)$ is randomly sampled for each domain $d_i \in D$ to obtain subsets $\varepsilon^*_T(d_i)$. These experts are concealed from the machine learning component of the framework, and a portion of ratings provided by them withheld. The remaining rated resources are then used to produce recommendations for each expert. Predictive accuracy metrics will then be used to obtain a threshold on recommendation confidence, such that any recommendations made above the threshold are likely to be *true positives*; i.e. the items removed from $\rho'(e_j)$, $e_j \in \varepsilon^*_T(d_i)$ dominate the recommendation lists produced. Once such thresholds have been obtained for each $d_i$, the automatically generated profiles are used to produce recommendations in each domain. The difference between the upper bound of confidence in the recommendations produced when using the generated profiles, and the domain thresholds obtained can then be assessed.

In addition, it is argued that the generated profiles can be restricted, based on the contextual setting at the time of recommendation. To assess this claim, different subsets of the same profile will be used to produce recommendations.

The experiments discussed above are carried out in Section 5.4

### 3.3.4 Overall recommendation quality

While the main focus of the framework is to be able to produce cross–domain recommendations, the overall quality of the recommendation made is an issue that warrants assessment. Discussion in the previous chapter has pointed out that predictive accuracy should not be considered as the single means of evaluating the performance of a recommender. This section provides a discussion on the metrics selected to evaluate experimental results in this dissertation.

**Popularity and Authority:** Popularity is defined as the ratio of experts that have rated a resource. In contrast Authority is a measure of the resource's centrality in the dataset – how well connected to other resources it is. The notion of Authority with respect to the probabilistic model presented in Chapter 5 is provided in Sections 5.2.1 and 5.3.1. As such the relationship between the two metrics can provide evidence on how well the model used corresponds to the preferences of domain experts.

**Ratability:** The probability that the user will provide a rating for a recommended resource. In the context of the framework described herein, this metric coincides with the value used to rank recommendations. Comparisons between these values can be used to establish whether good recommendations are made with similar confidence across different domains.

**Intra–list similaririty (ILSM):** The average similarity of any two resources that appear in a list of recommendations. This metric describes the diversity of recommendation lists. As resources in the proposed framework can be described either using domain–specific representations, or projections from the universal vocabulary, two definitions are provided: ILSM–RV and ILSM–WV respectively. The relationship between the two scores can be used to verify the appropriateness of the corpus used as the universal vocabulary.

**Rank:** When recommendations are drawn from the same domain as profiling features, a single resource is withheld from each profile used in the experiments presented in Section 5.3.1. This metric measures the rank of the withheld resource in the corresponding recommendation list, and can be used to assess the accuracy of recommendations.

**Profile rank:** The experiments presented in Section 5.3.2 make recommendations across multiple domains. This is done by identifying a set of common experts, and recommending resources from one domain, based on profiling information from another. This metric then gives the average rank in the list of recommendations produced in order to recover all the profiling elements in the recommendation domain.

**Top rank:** Related to the Profile rank metric, this is the resource that appears in the experts profile in the recommendation domain, and is ranked the closest to the top of the recommendation list.

It is noted that direct comparisons between the values of the Rank, Profile rank, and Top rank metrics cannot be made, as different selection effects are present. Regarding the performance of the framework, they represent the average–case, worst–case, and best–case scenaria respectively.

**Personalisation:** The degree to which recommendations differ from user to user, measured using Spearman's Footrule [82]. While the production of personalised recommendation lists is implicitly important to any recommender system, these values can also be used to detect changes in performance under different experimental set–ups.

## 3.4 Summary and conclusions

The terminology required to express the various elements of the recommendation framework was defined in this chapter, and used to present an overview of the recommendation process as dictated by the framework. Further, the experiments carried out to assess the various operations carried out during this process have been outlined. The following section revisits the problems identified in Section 2.1.8 and discusses how they are addressed by the framework presented herein.

### 3.4.1 Well known pitfalls in deploying RS revisited

- The 'cold start' problems

  Cold start problems are rendered irrelevant in the context of the proposed framework. This is the case since users are not assigned empty profiles upon registration, but rather carry with them the information they accessed or created in the past. Of course, if users have not created any information prior to subscribing with the system (or have chosen to not disclose any) the problem persists. However, such behaviour would somewhat defeat the point of seeking personalised recommendations. Similarly, resources for recommendation are imported only if a member of a community with expertise in resources of that type publishes their evaluation of it.

- The most similar items are not always good recommendations

  The fact that the framework requires each resource to be mapped to a unique set of terms in the universal vocabulary provides a mechanism for identifying interchangeable resources. Such resources are expected to have identical descriptions using terms from the universal vocabulary and can therefore be merged.

- Shifts and temporal cycles of user interests

  The rich representations used by the profiling component of the framework enable the segmentation of user profiles based on contextual attributes. Information created or accessed by a users during a specific time interval can easily be selected from their profile. As such, shifts of interest and temporal cycles can be accommodated for in the framework in two ways – either by considering only the most recent (subjective to a threshold) elements of a profile in order to make recommendations, or by requiring that users specify a period where their interests were highly related to their current recommendation need.

- Recommendations made independently of context

  Although connected to the previous point, it is argued here that temporal attributes alone are not adequate for representing the context of a recommendation. While the framework does not offer a solution for automatically determining which aspects of a user profile are relevant to the context of a particular recommendation, it provides a novel tool for assessing the effects of using profile segments explicitly selected to reflect a particular context.

- Only items described in one pre-specified representation are considered

  The notion of an exhaustive index of the resources to be recommended does not exist in this framework. Instead, any resources the user has implicitly expressed interest in form part of their profile, regardless of their origin. As such, the effects of problems associated with the inadequacy of user profiles to represent a wide range of user interests are expected to be less severe.

- Potential biasing effects

  The framework shifts the emphasis from increasing product sales, to satisfying user needs in deploying RS technologies. By importing preference data from external expert communities, it becomes harder to spuriously insert an arbitrary resource. Moreover, to influence the system to recommend the said resource over others, one would also have to obtain control over the universal representations of resources and the semantic connections between their descriptive terms in the universal vocabulary. Furthermore, since the external communities are simply seen as forums to publish the preferences of their members, there is no guarantee on which vendor will be selected by a user deciding to purchase a recommended resource.

# Chapter 4

# The Universal Vocabulary: Wikipedia

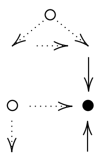


Fig. 1.2

In the previous chapter the difficulty in comparing heterogeneous external communities of domain experts to user profiles was identified. To deal with this issue, Def. 5 introduced the notion of a universal vocabulary. Provided that both the external communities and user profiles can be described using this vocabulary, the aforementioned comparison can be easily facilitated using the new representation. Of course, a vocabulary expressive enough to describe every conceivable resource can not be expected to be readily available. Thus, the possibility of using Wikipedia as an adequate replacement is explored in this chapter.

Wikipedia [87] was identified as a rich external source of information, due to its wide coverage of subjects, maintained by their respective interest groups [28]. There has been much discussion around the need for the adoption of a *lingua franca* for the successful deployment of semantic technologies and the impossibility of imposing a common, engineered understating of the world. Wikipedia, as the largest body of information that is freely accessible and consensually built, can be regarded as a prominent candidate to fill this gap. This information may be deemed expert knowledge, and the web–graph spanned by its articles together with the links between them can be used as the universal vocabulary in our approach. In addition, the fact that articles are organised in categories provides added opportunities for extracting some semantics on the quality of the matching carried out.

In order to project a resource onto Wikipedia we identify a page that corresponds to the resource. The projection then contains the aforementioned page along with pages that link to, or are linked from it. Thus, the function $\varphi$, defined in the previous chapter to map a resource $r_i$ to its universal description will give:

$$\varphi(r_i) = \{f_1, f_2, \cdots, f_n\} \tag{4.1}$$

The projection contains the Wikipedia page that corresponds to domain resource $r_i$, and any other pages connected to it via hypelinks.

In some cases, pointers to Wikipedia pages that correspond to domain resources are provided by external bodies such as the Musicbrainz knowledgebase [83] in the music domain. In others, such as the film domain, Wikipedia categories can be used to assess whether the page actually corresponds to the resource, or if it refers to a different concept with the same name. Both situations are described in detail in Section 4.3. The following section presents the approach taken when links between domain resources and Wikipedia articles cannot be found using the aforementioned techniques.

## 4.1 Problems

Wikipedia does not actually contain a page for each resource in the world and using it as the Universal Vocabulary implies that some resources cannot be represented. Therefore the framework would be unable to predict the utility of such resources to the user. In such cases it is assumed that the experts who rate the resources also provide textual descriptors of them.

Given the recent uptake of 'free–form tagging' by social networking sites, the assumption is not an unreasonable one to make. Colleagues working on the TAGora project have developed an approach, based on a sequential application of filters, for matching such tags to Wikipedia articles detailed in [15]. This process, summarised below, is used to obtain Wikipedia representations for resources described by tags.

All tags that appear in a domain are first collected and then processed as follows:

1. Lexical filtering

   First, tags that are a single character long are removed from the dataset, as are tags longer than 25 characters. Tags that contain numbers and fall under a global frequency threshold are also discarded. The rest are then processed to convert any 'special' characters that may occur to a base form (e.g. ö to o). Finally, common stop-words such as pronouns, articles, prepositions, and conjunctions are removed.

2. Compound nouns and misspellings

   The Google 'did you mean' feature provides an excellent way to resolve compound nouns and misspellings. Since this is based on the global frequencies of words on the Web, it is able to resolve common misspellings or abbreviations that would not appear in a standard dictionary. Compound nouns are also common in tags since most tagging systems do not allow spaces in tags. As such users resort to dropping spaces all together, or to using '-', '_', etc. as delimiters. Google is effective for splitting tags consisting of two terms, but is likely to fail for tags that consist of more. As such, the corpus of tags is iterated over, using the decomposed tags as prefixes and suffixes.

3. Wikipedia lookup

   Wikipedia is then queried for the filtered tags. In particular, a Wikipedia disambiguation page is requested for each tag first, and if one is not found but an article matching the tag exists (after following any redirects), the tag is considered to correspond to the article. If a disambiguation page is obtained, the tag is considered to have multiple meanings. For the purposes of this dissertation, tags with multiple meanings are ignored.

4. Morphologically similar terms

   A custom singularisation algorithm, together with the Snowball[51] stemming library are used to merge morphologically similar tags such as *blog, blogs* and *blogging.* The Wikipedia articles matched previously with tags are also associated with morphologically similar terms.

5. WordNet Synonyms

   Synonyms are often used to communicate a certain concept. As such WordNet synsets, [58], are used to merge together synonymous tags. As in the previous step, Wikipedia matches are shared amongst synonyms.

Any tags not matched to Wikipedia pages in this way are discarded by the recommendation framework. This is the case since the disambiguation of tags with multiple meanings is the object of current work within the TAGora project, and has not been available at the time the experiments were carried out. Moreover, while the filtering of 'special' characters in step 1 may be seen to increase tag polysemy, it is required to carry out the Wikipedia look-up step. While these limitations are acknowledged, the disambiguation step was considered beyond the scope of this dissertation.

Resources are then projected onto the recommendation space by aggregating the mappings obtained for tags associated with them. More precisely, considering $T$ as the set of all tags, a further two functions are defined, $\tau^*$ and $\tau$:

$$\begin{aligned} \tau \quad : \quad R \quad &\rightarrow \quad 2^T \\ r \quad &\mapsto \quad \tau(r) = \{t_1, t_2, \cdots, t_m\} \end{aligned} \tag{4.2}$$

$$\begin{array}{rcccl} \tau^* & : & D & \rightarrow & 2^T \\ & & d & \mapsto & \tau^*(d) = \bigcup_{r \in \rho^*(d)} \tau(r) \end{array} \tag{4.3}$$

As each tag can be considered a resource in its own right, $T \subset R$.

For each resource $r$, $\tau(r)$ is then the set of tags the resource has been associated with, while $\tau^*$ maps domain $d$ to the set of tags that have been used to describe resources in $\rho^*(d)$. Write:

$$\forall r \in \rho^*(d)\,,\, \tau(r) \subset \tau^*(d) \tag{4.4}$$

Since the set of all tags $T$ is a subset of $R$, the set of all resources, the function $\varphi$ can be applied to a tag to obtain its projection in Wikipedia, as for any other resource.

## 4.2 Recommendation space

While Wikipedia is considered adequate to play the Universal Vocabulary role, it is computationally infeasible to consider the entire corpus each time a recommendation is to be made. Instead we identify a recommendation space associated with each recommendation domain. This space, $S$, is a graph whose nodes are given by:

$$S_i = \varphi^*(\rho^*(d_i)) \tag{4.5}$$

Edges in the graph encode hyper–links between the Wikipedia articles that correspond to the nodes.

The space is represented using an adjacency matrix. This is a square matrix with a row and a column allocated for each element of $S_i$. The adjacency matrix contains binary entries indicating the presence of a hyperlink between the two corresponding articles in Wikipedia.

Once a recommendation space is extracted, we obtain a projection of the user in this space by identifying the elements in their profile that also appear in this space.

$$\varphi^*(\rho(u)) \cap \varphi^*(\rho^*(d_i)) \tag{4.6}$$

The remainder of this chapter is devoted to assessing the utility of this space as a basis for making recommendations.

## 4.3 Assessing the appropriateness of Wikipedia as the universal vocabulary

The experiments presented in this section share a common intuition. If Wikipedia is an adequate universal vocabulary, the projections of similar resources should remain similar. This chapter presented three ways of matching domain resources to Wikipedia articles. Here, we assess the performance of each one, in terms of retaining similarity between resources.

In each experiment, the corresponding dataset of domain resources is randomly sampled to obtain 1 000 resources, each one associated with a vector of ratings assigned to it. Similarity between each pair of resources is computed using the vector cosine similarity metric. An adjacency matrix is then constructed for the recommendation space, a subset of the web–graph spanned by Wikipedia. The rows of this matrix that correspond to the articles $r_i^w$ (and in turn to domain resources, Eq. 4.1) are used to calculate similarity between the same pairs. Vector cosine similarity values are obtained and compared to those obtained from the corresponding rating vectors.

### 4.3.1 When mappings between resources and articles are provided explicitly

In the music category, the Last.fm system records its users' music listening behaviour, and uses the data collected to drive additional services such as providing personalised radio stations and making recommendations. Instead of requiring users to explicitly rate the music they listen to, Last.fm records the number of times each piece of music has been listened to as an implicit indication of preference. The system uses URIs from the Musicbrainz knowledge base [83] as identifiers for the available artists. In turn, the Musicbrainz knowledge base can be exploited to obtain mappings between resources from the music domain and Wikipedia articles, as Wikipedia articles that correspond to recording artists are manually asserted into the KB.

The Last.fm dataset was first sampled randomly to obtain a set of 1 000 artists with Wikipedia pages. Using Definitions 3, and 4 each artist $r_i$ is associated with a vector:

$$\vec{\nu}_{r_i} = \langle \nu(e_1, r_i), \nu(e_2, r_i), \cdots, \nu(e_{|\varepsilon^*(d_L)|}, r_i) \rangle \tag{4.7}$$

containing the frequency of listening to particular pieces of music by the artist $r_i$ for each Last.fm user $e$, where $d_L$ denotes the Last.fm domain. We assess similarity between the pairs of resources $\{r_i, r_j\}_n$ by calculating the cosine of the angle between their corresponding rating vectors.

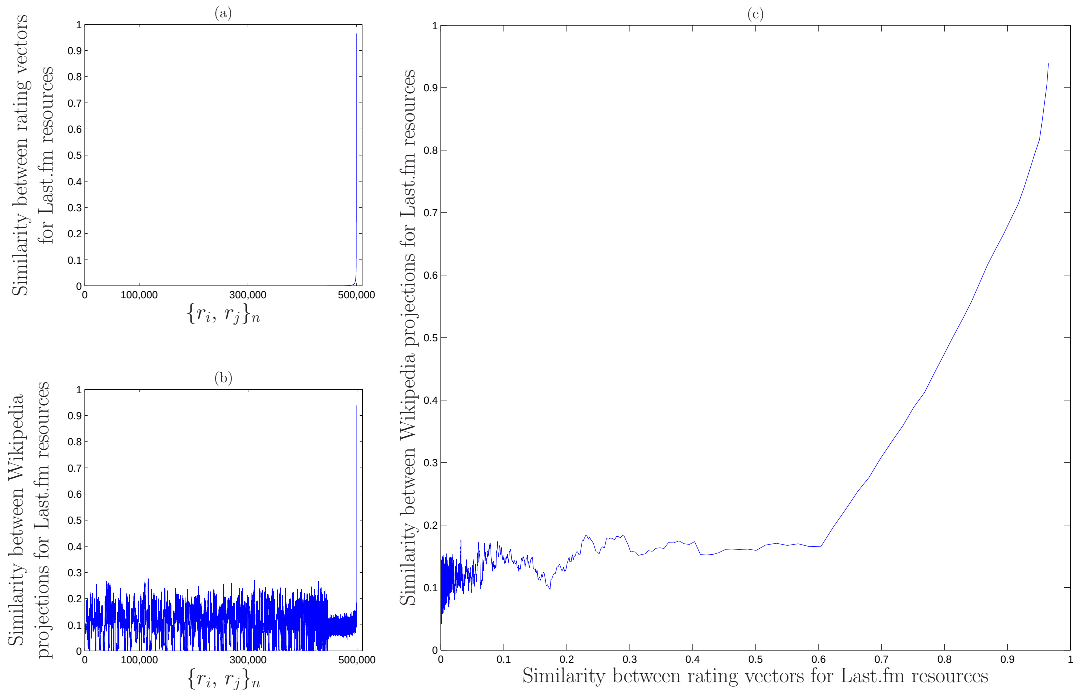


FIGURE 4.1: (a) The vector cosine similarity score of all pairs of vectors with Last.fm user ratings for the 1 000 artists selected for evaluation. Pairs are sorted according to their similarity score.
(b) The vector cosine similarity score for rows in the Wikipedia adjacency matrix that correspond to the same pairs of artists as in (a).
(c) Similarity of rows in the Wikipedia adjacency matrix as a function of the similarity of the rating vectors for the corresponding artists.

$$sim(\{r_i, r_j\}_n) = cos(\theta_n) = \frac{\vec{\nu}_{r_i} \cdot \vec{\nu}_{r_j}}{\|\vec{\nu}_{r_i}\| \; \|\vec{\nu}_{r_j}\|} \tag{4.8}$$

where $\|\vec{\nu}_{r_i}\|$ is the Euclidean norm of the vector $\vec{\nu}_{r_i}$.

Figure 4.1(a) shows the similarity values obtained for each of the 499 500 pairs, sorted in ascending order:

$$sim(\{r_i, r_j\}_1) \leq sim(\{r_i, r_j\}_2) \leq \ldots \leq sim(\{r_i, r_j\}_{499500}) \tag{4.9}$$

The graph illustrates the negative effects of data sparsity, as the vast majority of pairs are orthogonal. In fact, only 53 158 pairs give similarity values greater than 0.

A recommendation space has been constructed for the same 1 000 resources and encoded in an adjacency matrix. The vector cosine similarity metric is then applied to pairs of rows in the matrix that encode Wikipedia pages $r_i^w$ that correspond to the artists. Note that as defined in Definition 4, the function $\varphi$ would return the indices of non–zero entries in the rows of the adjacency matrix corresponding to its domain. For simplicity, it will be assumed here that the output of $\varphi$ will be the row corresponding to its input.

$$sim(\{\varphi(r_i), \varphi(r_j)\}_n) = cos(\theta'_n) = \frac{\vec{\varphi}(r_i)\vec{\varphi}(r_j)}{\|\vec{\varphi}(r_i)\|\|\vec{\varphi}(r_j)\|} \tag{4.10}$$

Again the operator $\| \cdot \|$ gives the Euclidean norm of a vector.

For comparison purposes the ordering of pairs is retained from above. A sliding window approach was employed to eliminate some noise and reveal the underlying trend. The width of the window has been set to 30, selected empirically to provide a detailed but clear illustration. Thus, the 30th pair will be assigned the average similarity of pairs 1–30 (inclusive), while the 31st will be assigned the average similarity of pairs 2–31. Figure 4.1(b) provides the resulting graph.

A first observation to make is that Figure 4.1(b) oscillates wildly for the segment where Figure 4.1(a) stays flat at 0. In other words, the Wikipedia representations of artists can be used to obtain a ranking in terms of similarity for pairs of resources that appear completely dissimilar when the rating vectors are used. However, the semantics of this observation are unclear. Further experimentation and analysis is required to assess whether people are more positively predisposed to listening to two artists with low similarity in terms of their Wikipedia representations, than to pairs of artists with an even lower similarity score.

Second, for pairs whose rating vectors have non–zero similarity, the two graphs exhibit very similar behaviour. That is, for any two pairs of artists in the sample $\{r_i, r_j\}$ and $\{r_{i'}, r_{j'}\}$ with:

$$sim(\{r_i, r_j\}) \geq sim(\{r_{i'}, r_{j'}\}) \geq 0.6 \tag{4.11}$$

Then:

$$sim(\{\varphi(r_i), \varphi(r_j)\}) \geq sim(\{\varphi(r_{i'}), \varphi(r_{j'})\}) > 0.15 \tag{4.12}$$

The relationship between the two sets of similarity scores is presented in Figure4.1(c), where the similarity between pairs of Wikipedia representations of resources is plotted as a function of the similarity of the corresponding rating vectors of the resources. Correlation between the two similarity scores can be seen clearly for pairs of artists whose rating vectors give rise to a similarity score greater than 0.6. While there are obvious disagreements between the two scores before this point, the curve has a slight, but positive gradient. Conversely, it is much more important for recommendation purposes to retain the ranking of highly similar resources than that of resources with low similarity scores which can be safely discarded.

### 4.3.2 When mappings between resources and articles are provided implicitly through the categories articles belong to

The aim of this dissertation is to detail a cross–domain recommendation framework that is dynamically extensible through importing new domains of resources for recommendation, along with expert preferences. This chapter presented the use of Wikipedia as common vocabulary to represent heterogeneous resources in a common format, allowing for comparisons to be made.

The previous section provided an assessment of using Wikipedia for this purpose, when high quality mappings between domain resources and Wikipedia articles are provided by an external source. However, compiling such mappings is a labour intensive process, and introducing this as a requirement is considered restrictive in term of the extensibility of the framework. Here, we assess whether the fact that Wikipedia articles are organised in categories can be exploited to automatically obtain such mappings.

The Netflix dataset was used in order to carry out this aspect of evaluation. Netflix provides an online DVD rental service to over 2 million subscribers. Users can rent movies they select out of the 17 770 titles that Netflix carries, and are later encouraged to provide ratings for the movies they watched.

Each title in the Netflix dataset is first sent to the Wikipedia search interface. Wikipedia ranks the search results in order of relevance to the query. If an article is returned with 100% relevance, that article is considered a candidate to represent the movie title that was searched for. In order to validate the match, the categories the article belongs to are obtained and processed.

In particular, if the article belongs to at least one category which contains the terms *'Film'*, *'Series'*, *'TV'*, *'DVD'*, *'Episode'*, or *'Show'*, then it is considered to be a good match. The process identifies Wikipedia articles representing 10 088 DVD titles. Once mappings have been obtained, the evaluation to assess their quality is carried out in the same manner as the previous section.

The mapped resources are first sampled to obtain a set of 1 000 movies. Figure 4.2(a) presents the similarity scores obtained for each pair of resources $\{r_i, r_j\}$ in this set, using the rating vectors

$$\vec{\nu}_{r_i} = \langle \nu(e_1, r_i), \nu(e_2, r_i), \cdots, \nu(e_{|\varepsilon^*(d_N)|}, r_i) \rangle \tag{4.13}$$

associated with resources in the Netflix dataset, $d_N$. Similarity between resources is given by Equation 4.8, and pairs are again sorted in ascending order of similarity. It can be seen from the graph that the level of sparsity is lower in Netflix than in Last.fm, as only a small number of pairs have no similarity.

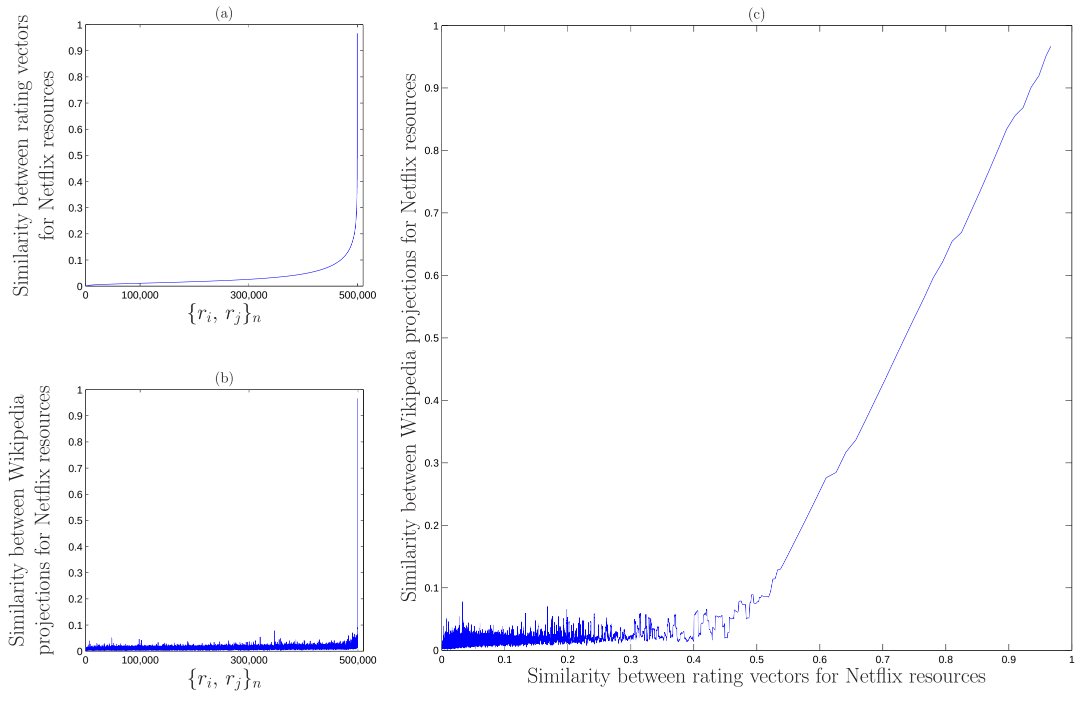


FIGURE 4.2: (a) The vector cosine similarity score of all pairs of vectors with Netflix user ratings for the 1 000 movies selected for evaluation. Pairs are sorted according to their similarity score.
(b) The vector cosine similarity score for rows in the Wikipedia adjacency matrix that correspond to the same pairs of movies as in (a).
(c) Similarity of rows in the Wikipedia adjacency matrix as a function of the similarity of the rating vectors for the corresponding movies.

Figure 4.2(b) presents the similarity score for the same ordering of pairs as for 4.2(a), when vectors $\vec{\varphi}(r_i)$, from the recommendation space are used. Similarity between such vectors is evaluated using Equation 4.10. It must be noted that although the overall shape of the curve may appear flat for the less similar pairs, it has a slight positive gradient.

Further, the fact that the curve of Figure 4.2(b) is much narrower than that of 4.1(b) provides some evidence to the claim that even for low similarity scores, the Wikipedia representations give rise to values proportional to those assigned by humans, based on taste. Since similarity using rating vectors is non–zero for almost all pairs, their arrangement in terms of similarity is more accurate. As such, the similarity scores for consecutive pairs, obtained using vectors from the recommendation space (the Wikipedia adjacency matrix) vary less. However, other factors may be involved, such as domain–specific attributes (e.g. movie articles could in general be less similar than those for recording artists) and thus the issue warrants further examination.

The correlation between the largest elements of the two sets of similarity scores is striking, as shown in Figure 4.2(c). When similarity between pairs of vectors in the recommendation space, $\varphi(r_i)$, is plotted as a function of the similarity of rating vectors, $\vec{\nu}_{r_i}$, for this sample, we find that:

If:

$$sim(\{r_i, r_j\}) \geq sim(\{r_{i'}, r_{j'}\}) \geq 0.52 \tag{4.14}$$

Then:

$$sim(\{\varphi(r_i), \varphi(r_j)\}) \geq sim(\{\varphi(r_{i'}), \varphi(r_{j'})\}) > 0.09 \tag{4.15}$$

### 4.3.3 When mappings between resources and articles are provided through processing tags associated with the resource

This section revisits the difficulties identified in Section 4.1, when mappings between resources and Wikipedia articles are neither provided explicitly, nor can domain resources be expected to fall under a small number of Wikipedia categories. Provided that domain experts are allowed to associate descriptive tags with the resources they evaluate, the opportunity arises to first map individual tags to Wikipedia articles. The process through which this is carried out was described in Section 4.1.

The del.icio.us social bookmarking site has been selected to demonstrate this approach. As subscribers are allowed to bookmark any URL they want, it may not be possible to match such resources to Wikipedia using the two aforementioned techniques. As such, resources are projected onto the recommendation space by aggregating the mappings obtained for tags associated with them:

$$\varphi(r) = \bigcup_{t \in \tau(r) \neq \emptyset} \varphi(t) \tag{4.16}$$

It must also be noted the the notion of ratings is absent in del.icio.us; either a user has bookmarked a URL, or not. Thus, the vectors

$$\vec{\nu}_{r_i} = \langle \nu(e_1, r_i), \nu(e_2, r_i), \cdots, \nu(e_{|\varepsilon^*(d_D)|}, r_i) \rangle \tag{4.17}$$

corresponding to URLs have Boolean entries, where $d_D$ is the del.icio.us domain.

The dataset is sampled randomly, as before, to obtain a set of 1 000 resources. However, here each URL is associated with two additional vectors: a tag vector

$$\vec{\tau}_{r_i} = \langle \tau(e_1, r_i), \tau(e_2, r_i), \cdots, \tau(e_{|\varepsilon^*(d_L)|}, r_i) \rangle \tag{4.18}$$

and a vector from the recommendation space, $\vec{\phi}(r_i)$.

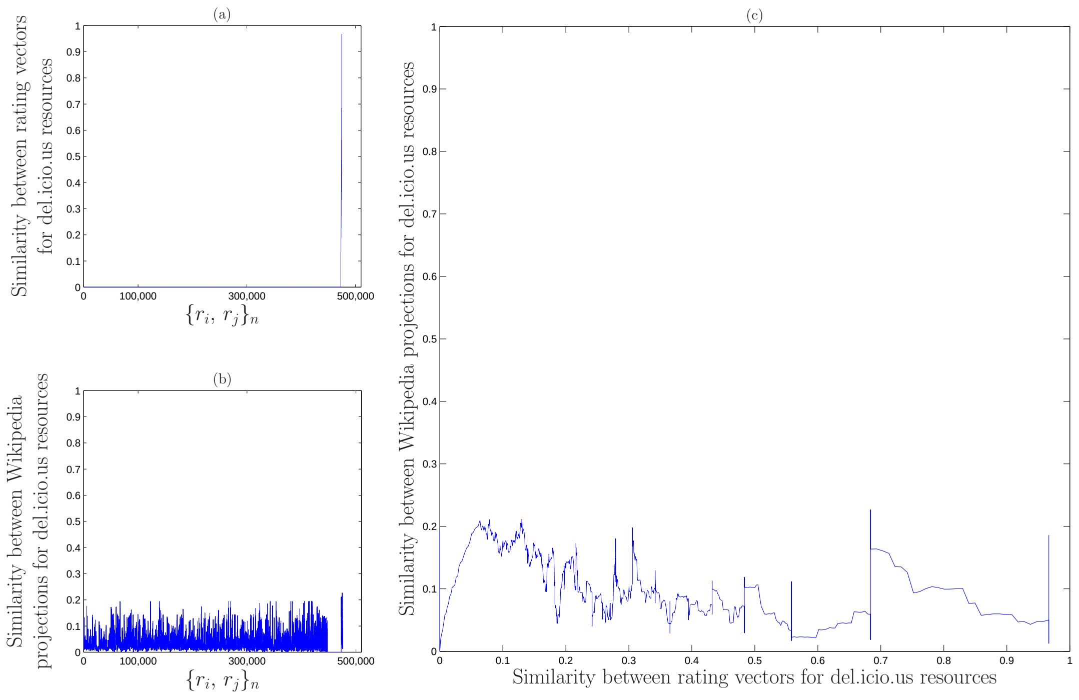


FIGURE 4.3: (a) The vector cosine similarity score of all pairs of vectors with Boolean entries indicating the del.icio.us users that have bookmarked the 1 000 URLs selected for evaluation. Pairs are sorted according to their similarity score.
(b) The vector cosine similarity score for rows in the Wikipedia adjacency matrix that correspond to the same pairs of URLs as in (a).
(c) Similarity of rows in the Wikipedia adjacency matrix as a function of the similarity of the vectors of (a).

Figure 4.3 provides an analysis of the relationship between the similarity of pairs of resources when they are expressed as rating vectors, and the similarity of the same pairs, seen as vectors from the recommendation space. When compared to Figures 4.1 and 4.2, the results in this domain are poor as the ranking of pairs of resources based on the similarity of their rating vectors appears to have no relationship to the similarity of their projections in Wikipedia.

With respect to the issue of ignoring polysemous tags, it is noted that it has no effect in the similarity between rating vectors (Figure 4.3(a))) as these vectors only contain the indices of users that have bookmarked a particular URL. Moreover, polysemous tags are also included in calculating similarity between URLs when expressed as tag vectors, $\vec{\tau}(r_i)$ (Figures 4.4(a), 4.5(b)). In fact, they have only been discarded from the vectors $\vec{\varphi}(r_i)$, as we have not attempted to map them to Wikipedia pages. However, Figure 4.4 shows a clear correlation between the similarity of tag vectors corresponding to URLs, and the similarity of vectors that contain the projection of tags with only one meaning in Wikipedia.

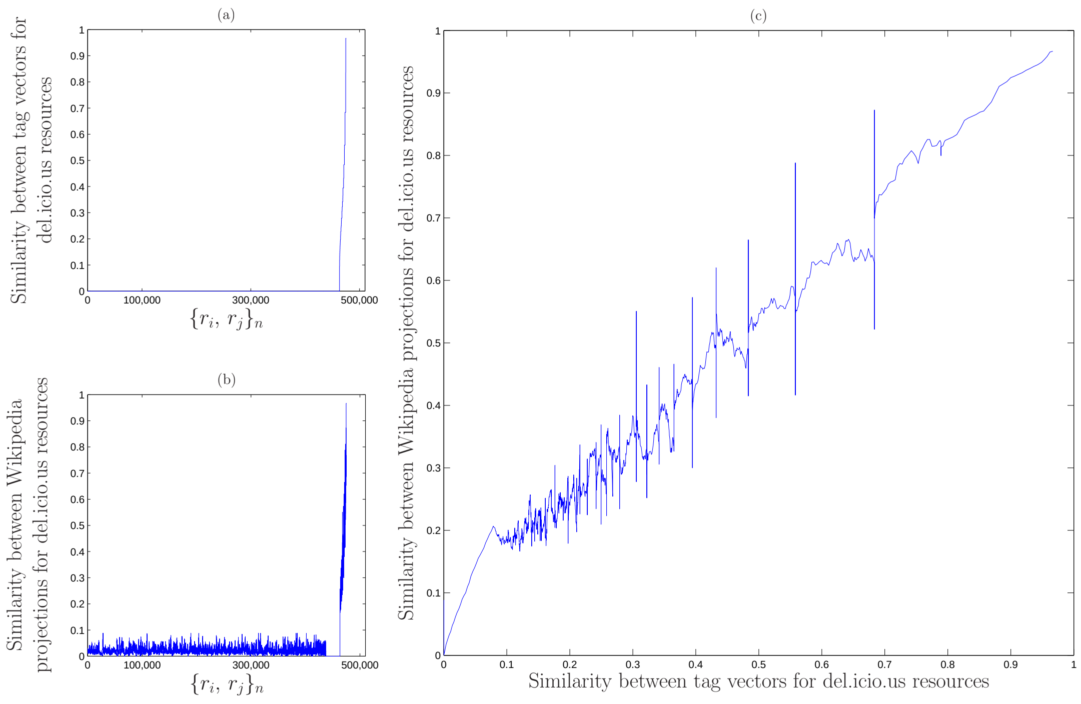


FIGURE 4.4: (a) The vector cosine similarity score of all pairs of vectors with Boolean entries indicating the tags associated with the 1 000 URLs selected for evaluation. Pairs are sorted according to their similarity score.
(b) The vector cosine similarity score for rows in the Wikipedia adjacency matrix that correspond to the same pairs of URLs as in (a).
(c) Similarity of rows in the Wikipedia adjacency matrix as a function of the similarity of the vectors of (a).

Motivated by this negative result, and since tags had been used as a proxy for bookmarked URLs to be projected onto Wikipedia, we examine the relationship between the vectors $\vec{\tau}(r_i)$ and $\vec{\varphi}(r_i)$.

Figure 4.4 indicates a very strong correlation between the similarity of vectors from the recommendation space and that of tag vectors, corresponding to the same pairs of resources. However, both definitions of similarity are less prone to the adverse effects of data sparsity in terms of user ratings than the Collaborative Filtering metric. First, this type of sparsity is irrelevant to the similarity of vectors $\vec{\varphi}(r_i)$, as rating information is not used. Second, as each expert in $e^*(d_D)$ provides at least one tag for each URL they bookmark, more tags can be found to correspond to a single URL than experts.

However the question whether vectors of tags corresponding to URLs convey the same kind of similarity between them as the fact that they have been bookmarked by the same users remains. The issue is addressed by performing the same type of analysis between the similarity of rating vectors representing resources, and that of corresponding vectors of tags. The results presented in Figure 4.5 show no evidence that they do. These results

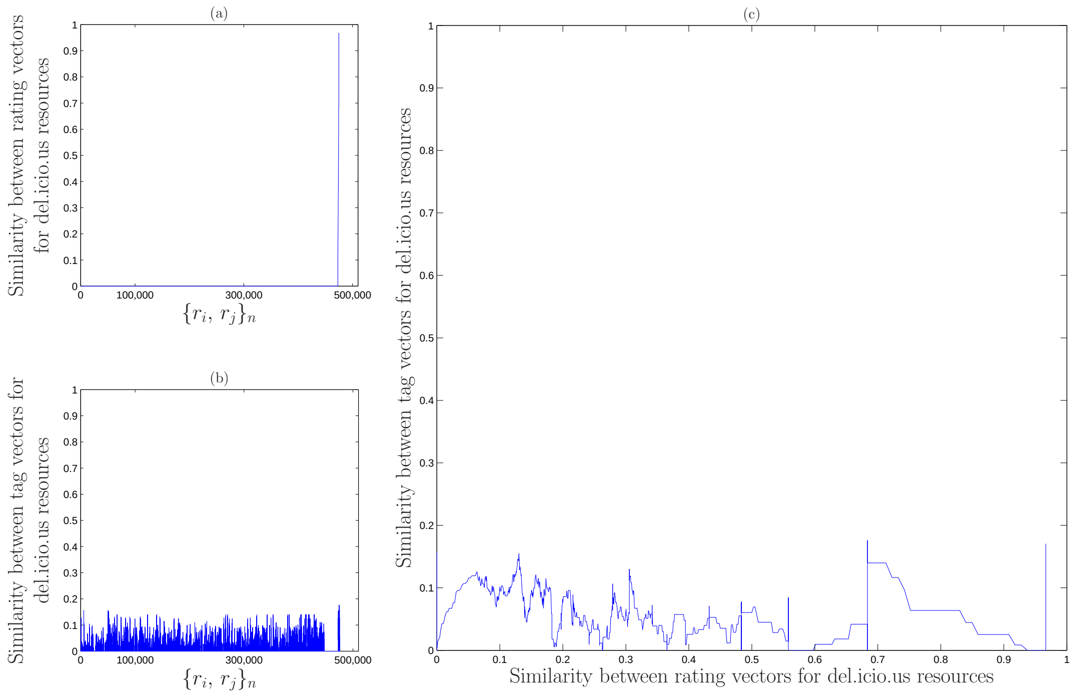


FIGURE 4.5: (a) The vector cosine similarity score of all pairs of vectors with Boolean entries indicating the del.icio.us users that have bookmarked the 1 000 URLs selected for evaluation. Pairs are sorted according to their similarity score.
(b) The vector cosine similarity score for vectors with Boolean entries indicating the tags associated with the same pairs of URLs as in (a).
(c) Similarity of vectors with Boolean entries indicating the tags associated with the bookmarked URLs as a function of the similarity of the vectors of (a).

would appear to suggest that even when the same URLs are bookmarked by different users, the tags used to describe them differ.

As eluded to earlier, rating vectors are more sparse than the other two types of vectors associated with resources, rendering the similarity metric less sensitive. Such a failure could eliminate any evidence of correlation, as pairs would not be sorted in the correct order. In fact, over 76% of the URLs have only been bookmarked by just one person, and only 1 216 of the total 598 660 URLs are bookmarked by more than 25 users. For comparison, only 4% of the Last.fm resources have been rated by only one person, while for Netflix each movie is rated by at least 3 subscribers.

To test whether any correlations between the similarity of rating vectors and that of corresponding vectors from the recommendation space become more evident as the sparsity of the rating vectors is reduced, more samples are obtained from the dataset. Instead of sampling the dataset randomly, a threshold in the number of del.icio.us subscribers that have bookmarked each URL in the sample is applied. Figures 4.6(a) – (d) provide the relationship between the two types of similarity for samples obtained at thresholds of 0, 25, 50, and 100 respectively.

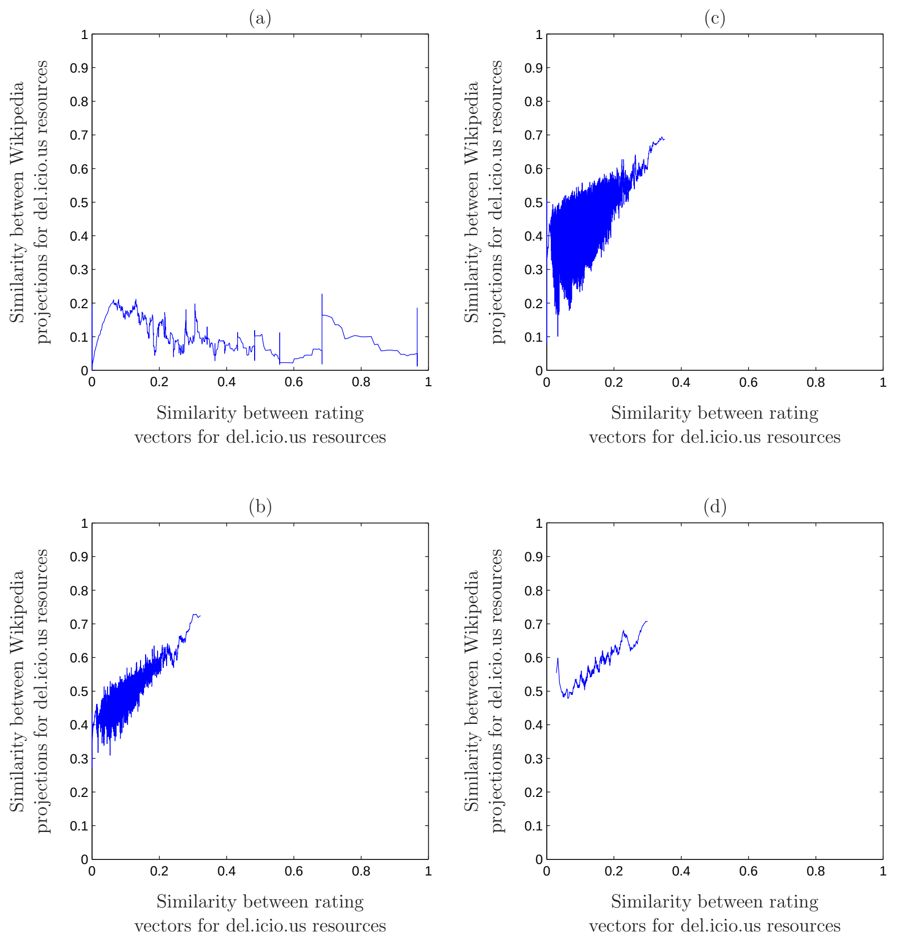


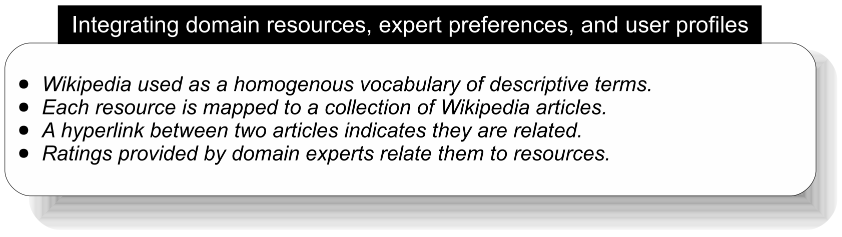
FIGURE 4.6: Similarity of pairs of rows in the Wikipedia adjacency matrix as a function of the vector cosine similarity score of corresponding pairs of vectors with Boolean entries indicating the del.icio.us users that have bookmarked:
(a) the 1 000 URLs randomly selected for evaluation,
(b) the 1 216 URLs bookmarked by more than 25 experts,
(c) the 277 URLs bookmarked by more than 50 experts,
(d) the 44 URLs bookmarked by more than 100 experts.

Indeed, some evidence of correlation is already recovered after the application of the lowest threshold, requiring that each vector contains at least 26 entries, as shown in Figure 4.6(b). Figures 4.6(c) – (d) show that this correlation grows stronger, as the rating vectors become more dense. In other words, as the similarity metric between rating vectors becomes more sensitive, and the sorting of pairs of resources based on it more accurate, the similarity scores between such pairs are found to correspond better with those obtained by comparing vectors that encode the web–graph spanned by articles in Wikipedia that correspond to the same pairs of resources.

The purpose of this section was to assess whether the projections of resources in the recommendation space, obtained through mapping descriptive tags to Wikipedia articles, can be compared to convey the same kind of similarity between resources as that obtained by comparing vectors of ratings. It has been shown that when the data is relatively dense this appears to be the case.

A further observation to make is that the upper bound in similarity between rating vectors appears to decrease, as the data becomes more dense. To re–iterate, resources expressed as rating vectors can be expected to become less similar, as more users rate them. In contrast, with the exception of Figure 4.6(a), the upper bound in similarity between vectors in the Wikipedia adjacency matrix appears to remain stable. This is interpreted as further evidence towards speculations voiced in the preceding two sections, that the ranking of two pairs of resources based on the similarity of their corresponding vectors in the recommendation space is reliable even in cases where both pairs receive low similarity scores.

## 4.4 Conclusions


Integrating domain resources, expert preferences, and user profiles

- *Wikipedia used as a homogenous vocabulary of descriptive terms.*
- *Each resource is mapped to a collection of Wikipedia articles.*
- *A hyperlink between two articles indicates they are related.*
- *Ratings provided by domain experts relate them to resources.*

FIGURE 4.7: A visual representation of the contributions in this Chapter.

The previous chapter set a requirement for a universal vocabulary to facilitate comparisons between heterogeneous resources from different domains, in order to deploy the cross-domain recommendation framework. This chapter proposed that Wikipedia can be used for this purpose and described methods to project domain resources onto Wikipedia, specific to domain characteristics.

Wikipedia articles corresponding to domain resources are identified in three ways: either by explicitly provided high quality mappings, or implicitly by issuing textual queries to Wikipedia while at the same time requiring that mappings belong to a specific set of Wikipedia categories related to the domain in question. When both these strategies are inapplicable, descriptive tags assigned to resources by domain experts can be mapped to Wikipedia articles, and in turn every resource to the articles corresponding to the tags assigned to it. It is assumed that resources from a comprehensive subset of available domains, as defined in Definition 4, can be mapped to Wikipedia in one of the three aforementioned ways.

Once domain resources are mapped to Wikipedia articles, a recommendation space is obtained by constructing a graph that contains nodes for each such article, and each other page linked to it. Edges then represent hyperlinks between Wikipedia articles. This space is encoded in a square adjacency matrix, with a row and column for each node in the graph with Boolean entries. An entry in this matrix represents an edge between the node encoded by the row index and that encoded by the column index.

To assess whether vectors from this matrix can be compared to convey the same kind of similarity between resources as that obtained by comparing vectors of ratings, a domain for which each method would be applicable was obtained and the resources associated with it processed. It was found that when similarity between rating vectors could be expected to be trustworthy (i.e. when data sparsity was low enough) the two similarity values appear to correlate.

For recommendation purposes it would suffice that highly similar pairs could be ranked in the same order based on either definition of similarity. However, evidence was also collected towards the claim that the ranking of pairs of resources based on the similarity of corresponding vectors from the recommendation space can be expected to be reliable even for pairs with very low similarity.

In Section 4.3.1 we found that pairs with low similarity scores in terms of rating vectors scored lower, on average, in terms of Wikipedia projections than pairs with orthogonal rating vectors. Considering that no sensible ordering based on cosine similarity can be imposed to orthogonal vectors, lead to the speculation that such pairs could be better ranked using the Wikipedia projection.

Using the denser Netflix dataset in Section 4.3.2, it was found that similarity between vectors from the recommendation space varied a lot less, even for extremely low values of similarity, than for the aforementioned orthogonal ones.

Finally, in Section 4.3.3, where the vast majority of rating vectors were orthogonal, we were unable to detect any correlation between the two definitions of similarity when pairs of resources were obtained through random sampling. However, by selecting samples that were denser, and thus obtaining a more reliable ranking of pairs, the correlations were recovered.

Although these effects could be attributed to factors that were not studied by the experiments presented here, the fact that they have been observed using three independent sets of data lends some credibility to the claim.

In summary, the experiments carried out in this chapter have found Wikipedia to be adequate for use as the universal vocabulary in our approach, with respect to the three domains that have been considered. In doing so, a methodology has been provided to assess the appropriateness of any corpus for this role. Other options would include WordNet [58], various library classification schemes, or specifically designed ontologies. The selection should in turn be guided by the characteristics of the recommendation domains considered, and the quality of projections obtained from this space subsequently assessed.

# Chapter 5

# A probabilistic framework

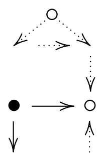


Fig. 1.2

This chapter provides the specifics of the machine learning component of the recommendation framework presented in this dissertation. Chapter 3 identified the availability of a universal vocabulary of terms and relationships between them as a requirement for a system to be able to recommend resources from domains different to those of profiling elements. Subsequently, the use of Wikipedia for this purpose was proposed in the previous chapter, where a number of ways to identify articles representative of domain resources have been presented. Wikipedia projections of domain resources were then obtained by compiling vectors encoding the hyperlinks contained in such articles. The experiments performed in Section 4.3 showed a correlation between the similarity of the projected representations and that of rating vectors corresponding to resources. In other words, domain resources can be expected to be rated by the same group of experts if their Wikipedia projections are similar. Moreover, the correlation was found to be evident in three independent domains.

The universal vocabulary can naturally be represented as a graph, since the definition places emphasis on the semantic relationships between terms. Moreover, the projected vectors are obtained by first compiling an adjacency matrix of the web–graph spanned in Wikipedia by articles that correspond to domain resources. Thus, the use of a graph–based algorithm was considered appropriate for the task of assessing resources for recommendation. Markov chains [50] provide a natural, yet very powerful way to model the traversal of edges in a graph. To do so, each edge is associated with the conditional probability of reaching the destination node given the origin.

The following section presents a number of pre–processing steps carried out in order to integrate the preferences expressed by domain experts in the same graph as the aforementioned projected representations, and Section 5.2 then provides details of the probabilistic model. The three available datasets are then used to evaluate the algorithm's performance in producing cross–domain recommendation. To carry this out, we first establish a baseline behaviour by using the algorithm to produce recommendations from the same domain as profiling features. Then, the performance of the algorithm in producing cross–domain recommendations is assessed in comparison to the baseline behaviour. Finally, the use of profiles generated by the Semantic Logger system to produce recommendations is demonstrated in Section 5.4

## 5.1 Data pre–processing

The datasets RS systems typically deal with are both incredibly large, and vastly sparse. To illustrate, we provide the characteristics of the three datasets chosen to carry out the evaluation of the framework in Table 5.1. The density column refers to the percentage of $\{expert, resource\}$ pairs where ratings are available.

| | Number of experts | Number of resources | Number of ratings | Density (%) |
|---|---|---|---|---|
| NetFlix | 2 649 429 | 17 770 | 100 462 465 | 0.21 |
| Last.fm | 810 104 | 141 078 | 2 572 868 | 0.0023 |
| del.icio.us | 1 965 | 598 660/27 027 (bookmarks/tags) | 877 599/273 357 (bookmarks/tags) | 0.075/0.52 (bookmarks/tags) |

Table 5.1: Characteristics of the Last.fm, NetFlix, and del.icio.us datasets.

The sheer size of the datasets introduces difficulties in merely loading the complete data in memory. As such, the use of complex, computationally intensive algorithms on the raw data is prohibited, since this would render the system unusable in terms of responsiveness. In addition, the extreme levels of sparsity could have adverse effects on the effectiveness of the framework. The following sections provide details of the various data pre–processing steps carried out to overcome such issues.

### 5.1.1 Sampling

First, the dataset is sampled in order to obtain a subset of resources and experts for which normalisation and summarisation can feasibly be carried out. It is important, however, that this sampling process does not alter the intrinsic structure of the ratings matrix. In order to ensure this, a number of statistics are preserved in the sample. The reader is first reminded of the following functions, defined in the previous chapter:

| | | |
|---|---|---|
| $\varepsilon(r)$ | : | The set of experts who have rated resource $r$. |
| $\rho^*(d)$ | : | The set of all resources in domain $d$. |
| $\varepsilon^*(d)$ | : | The set of all experts who have rated resources in domain $d$. |

| | | |
|---|---|---|
| $\rho'(e)$ | : | The set of resources rated by expert $e$. |

Let $d^{(n)}$ be a subset of $\rho^*(d)$ of size $n$. In particular there are $|\rho^*(d)|$ such singleton sets $d^{(1)}$, and one set $d^{(|\rho^*(d)|)}$. Thus, if $d'$ is one of the sets

$$d^{(k)}\,,\, k = 1, \dots, |\rho^*(d)| \tag{5.1}$$

sampled from domain $d$

$$\rho^*(d') \subset \rho^*(d) \tag{5.2}$$

Similarly

$$\varepsilon^*(d') \subset \varepsilon^*(d) \tag{5.3}$$

The relationship between a domain $d$ and a sample $d'$ drawn from it will be denoted $d' \subset d$ for convenience. The computational facilities available will dictate the sample size, and subsets from $\rho^*(d)$ and $\varepsilon^*(d)$ are initially drawn randomly. The following statistics are then checked to ensure that the sample retains the intrinsic structure of the original space:

1. The average number of ratings for each resource.

$$\frac{\sum\limits_{r \in \rho^*(d)} |\varepsilon(r)|}{|\varepsilon^*(d)|} = \frac{\sum\limits_{r' \in \rho^*(d')} |\rho(r')|}{|\rho^*(d')|}, \quad \forall d' \subset d \tag{5.4}$$

2. The average number of ratings for each expert.

$$\frac{\sum\limits_{e \in \varepsilon^*(d)} |\rho'(e)|}{|\varepsilon^*(d)|} = \frac{\sum\limits_{e' \in \varepsilon^*(d')} |\rho'(e')|}{|\varepsilon^*(d')|}, \quad \forall\, d' \subset d \tag{5.5}$$

3. The density of the dataset.

$$\frac{\sum\limits_{r \in \rho^*(d)} |\varepsilon(r)|}{|\varepsilon^*(d)||\rho^*(d)|} = \frac{\sum\limits_{r' \in \rho^*(d')} |\varepsilon(r')|}{|\varepsilon^*(d')||\rho^*(d')|}, \quad \forall\, d' \subset d \tag{5.6}$$

The process is repeated until a sample of the intended size that satisfies the requirements posed above is obtained.

### 5.1.2 Normalisation

Absolute ratings cannot carry objective interpretations. For example, consider two experts, $e_1$ and $e_2$, who have both rated the same two resources. Assume that $\rho'(e_1) = \rho'(e_2) = \{r_1, r_2\}$, and that the experts provided the following ratings:

$$\nu(e_1, r_1) = 1 \qquad \nu(e_2, r_1) = 2$$
$$\nu(e_1, r_2) = 2 \qquad \nu(e_2, r_2) = 4$$

It would be a mistake to proclaim that $e_2$ values both resources twice as much as $e_1$. Instead, it is likely to be the case that both experts value the second resource twice as much as the first one. Therefore, for each expert we normalise $|\rho'(e_i)|$ as follows:

$$norm(\nu(e_i, r_j)) = \frac{\nu(e_i, r_j) - mean\left(\bigcup_{r' \in \rho'(e_i)} \nu(e_i, r')\right)}{std\left(\bigcup_{r' \in \rho'(e_i)} \nu(e_i, r')\right)} \tag{5.7}$$

where the functions $mean()$ and $std()$ give the mean and the (sample) standard deviation of a set, respectively. The normalised ratings are then standard scores (or $z$–scores), widely used in statistics to indicate how many standard deviations an observation is above or below the mean. Such scores are useful as they allow the comparison of observations from different normal distributions. In this case the normalised ratings allow for a more objective comparison between ratings provided for the same resources, by different experts.

Returning to our earlier example we obtain:

$$norm(\nu(e_1, r_1)) = -\frac{1}{\sqrt{2}} \qquad norm(\nu(e_2, r_1)) = -\frac{1}{\sqrt{2}}$$
$$norm(\nu(e_1, r_2)) = \frac{1}{\sqrt{2}} \qquad norm(\nu(e_2, r_2)) = \frac{1}{\sqrt{2}}$$

reflective of the fact that both experts opinions regarding each of the resources are relatively the same.

### 5.1.3 Summarisation

Having reduced the dataset to a size which we are able to process, we now have to deal with the sparsity issue. Sparsity is a problem, because if the overlap between the items that the experts have rated is not sufficient, the similarity metrics used can be rendered ineffective. Computationally cheap summarisation methods can be applied to the dataset in order to merge similar users together, reducing the overall sparsity of the dataset. The *mrkd–tree* (Multi–resolution k–dimensional tree)[21] was selected as an appropriate summarisation method, since it has been successfully deployed in accelerating other machine learning algorithms, such as Expectation Maximisation [91].

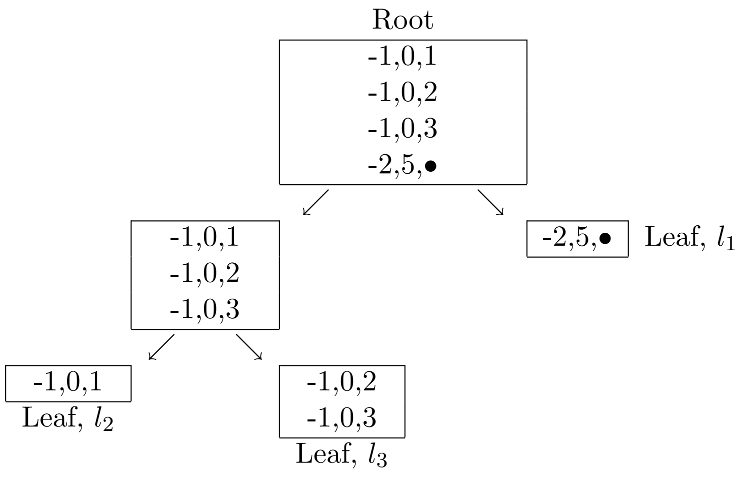
The mrkd–tree is a binary tree where each node contains a subset of the data points (which in our case correspond to the experts). All points belong to the root node and each node has two children that split the parent's data points between them. The tree is then built top-down, by identifying the widest dimension of the bounding hyper-rectangle at the current node and splitting its contents in two using the middle value. This is the dimension where the dataset varies the most, at the current node. Any points with entries in the selected dimension strictly lower than this middle value are assigned to the left child node and the rest to the right one. A maximum value for the widest dimension is defined *a priori*, and once the widest dimension of the datapoints contained in a node becomes lower than that, the node is not split further and becomes a *leaf node*.

To illustrate consider the following example of building an *mrkd*–tree for a dataset of 4 experts and 3 resources :

| | $r_1$ | $r_2$ | $r_3$ |
|---|---|---|---|
| $e_1$ | -1 | 0 | 1 |
| $e_2$ | -1 | 0 | 2 |
| $e_3$ | -1 | 0 | 3 |
| $e_4$ | -2 | 5 | ● |

Where ● stands for a missing value: the fact the expert $e_4$ has not rated resource $r_3$. All experts belong to the root node of the tree, with the widest dimension being resource $r_2$. That is to say the minimum and maximum ratings received by that resource differ the most. Thus the root node will be split along this dimension at the middle value $\frac{0+5}{2} = 2.5$. Any experts who provided a rating for $r_2$ strictly lower than 2.5 will be assigned to the left child node and the remaining ones to the right. Now the right child node is a leaf, as it only contains one expert and cannot be split further. The widest dimension in the left child node is for resource $r_3$, with width $3 - 1 = 2$. Assuming the threshold for the widest dimension is set to 1, this node will be split further based on the rating provided by the remaining 3 experts for resource $r_3$. The middle value is $\frac{1+3}{2} = 2$.

Although the intuitive, recursive definition of the tree–building process runs in exponential time, an iterative version of the algorithm can be defined that terminates in linear time. Once the tree is built, we discard the structure and only record the leaf nodes. Moreover, the normalised ratings are replaced with the original ratings provided by the experts.



$$l_1 = \{e_4\} \tag{5.8}$$

$$l_2 = \{e_1\} \tag{5.9}$$

$$l_3 = \{e_2, e_3\} \tag{5.10}$$

Leaves are thus obtained by aggregating a number of experts. In Definition 3 in the previous chapter, the function $\varepsilon(r)$ was introduced to map a resource to the experts that rated it. In order to describe the relationship between a leaf node and the resources rated by the experts contained within it, the same function will be used here to return the set of leaves with experts who rated resource $r$, as the notion of experts is replaced by that of leaves. Returning to the example provided above:

$$\varepsilon(r_3) = \{l_2, l_3\} \tag{5.11}$$

## 5.2 A Markov chain model

A graph–like structure provides a natural representation for the recommendation spaces considered by the framework. For a domain $d$, the graph encodes the link structure of Wikipedia for the elements of $\varphi^*(\rho^*(d))$. A node representing each leaf of the *mrkd–tree* is also included in the graph, with edges between leaf nodes and Wikipedia pages representing resources in $\rho^*(d)$, encoding ratings. Figure 5.1 gives an example of such graphs.

A Markov chain is used to represent the conditional probabilities, $P(A|B)$, of traversing an edge from node $B$ to node $A$. The probabilities are calculated as follows:

- Each node representing a leaf of the *mrkd–tree* receives a prior probability $P(l_k)$, equal to the number of experts contained in the leaf, normalised over the total

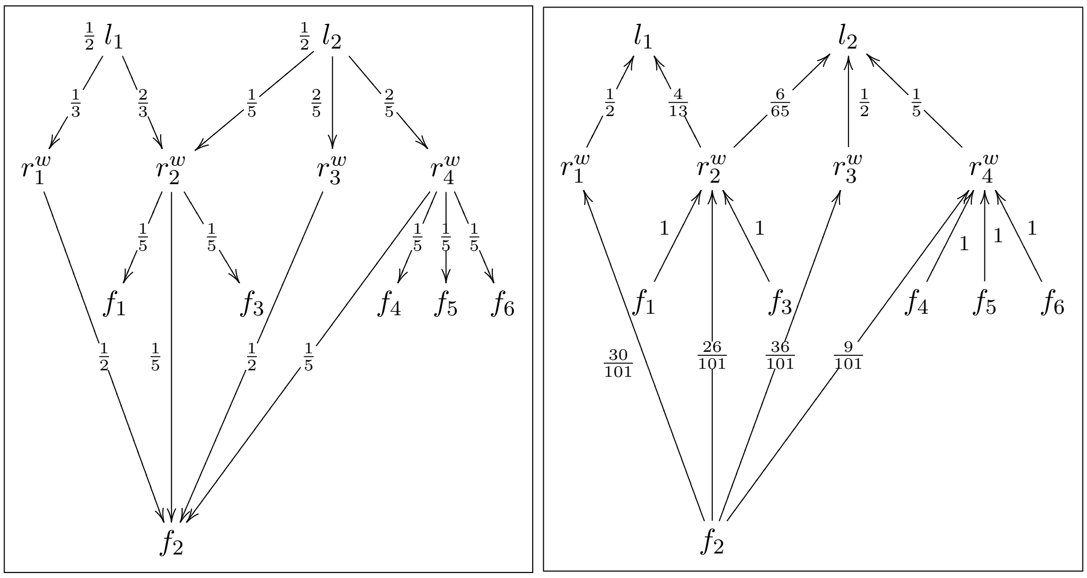


FIGURE 5.1: The two graphs present an example reversible Markov chain used by the framework. For clarity of presentation each graph gives the conditional probabilities associated with traversing edges in one direction. The model is then given by the integration of both illustrations.

number of experts in the domain.

$$P(l_k) = \frac{|l_k|}{|\varepsilon^*(d)|} \tag{5.12}$$

- Edges between leaves and Wikipedia pages representing resources receive conditional probabilities $P(r_i^w|l_k)$, proportional to the ratings provided by the experts. This is obtained by dividing each rating with the sum of ratings provided by the same expert, averaged over all experts in each leaf.

$$P(r_i^w|l_k) = \frac{\displaystyle\sum_{e \in l_k} \frac{\nu(e, r_i)}{\displaystyle\sum_{r' \in \rho^*(d)} \nu(e, r_i)}}{|l_k|} \tag{5.13}$$

- Edges connecting nodes that represent domain resources and other Wikipedia pages are traversed with equal probability. Using the $r_i^w$ nodes as a starting point the graph can be traversed towards either a leaf node, or a node corresponding to a Wikipedia page. So, in order to maintain that given a node the sum of conditional probabilities of all its adjacent nodes is equal to 1, the probabilities $P(l_k|r_i^w)$ and $P(f_j|r_i^w)$ are scaled by $P(leaf|r_i^w)$ and $P(link|r_i^w)$ respectively. These are the probabilities that starting from $r_i^w$ the graph is traversed to land on a leaf node, or a Wikipedia node and are given by:

$$P(leaf|r_i^w) = \frac{|\varepsilon(r_i^w)|}{|\varepsilon(r_i^w)| + |\varphi(r_i^w)|} \tag{5.14}$$

$$P(link|r_i^w) = \frac{|\varphi(r_i^w)|}{|\varepsilon(r_i^w)| + |\varphi(r_i^w)|} \tag{5.15}$$

Where the function $\varepsilon$ maps a resource to leaves of the *mrkd*–tree that contain ratings for it, and $\varphi$ returns all other Wikipedia pages connected via hyperlinks to the article mapped to the resource. By definition:

$$P(leaf|r_i^w) = 1 - P(link|r_i^w) \tag{5.16}$$

Now, the probability of traversing an edge from domain resource $r_i^w$ towards a connected article $f_j \in \varphi(r_i^w)$ is given by:

$$P(f_j|r_i^w) = \frac{P(link|r_i^w)}{|\varphi(r_i^w)|} \tag{5.17}$$

$$= \frac{1}{\text{Total number of nodes adjacent to } r_i^w} \tag{5.18}$$

Thus, resources are assessed for recommendation in this framework by evaluating :

$$\boxed{P(r_i^w|\varphi^*(\rho(u))) = \frac{P(\varphi^*(\rho(u))|r_i^w)P(r_i^w)}{P(\varphi^*(\rho(u)))}} \tag{5.19}$$

That is the probability of traversing the graph to reach node $r_i^w$ (representing resource $r_i$), using the Wikipedia nodes mapped to elements of user profiles given by $\varphi^*(\rho(u))$ as a starting point. To calculate this, one must provide definitions for the prior probabilities associated with resources $P(r_i^w)$, the marginal probability of observing the user's profile $P(\varphi^*(\rho(u)))$, and the likelihood $P(\varphi^*(\rho(u))|r_i^w)$ – the probability of traversing the graph towards the nodes that appear in the profile, using the node representing the resource being assessed for recommendation as a starting point. First, the graph is encoded in a probabilistic version of the adjacency matrix. Each cell in this matrix encodes for $P(row|column)$, the probability of traversing an edge from the column node to the row node.

The probabilities of traversing edges from nodes representing Wikipedia pages towards resource nodes $P(r_i^w|f_j)$, and from resources towards leaf nodes $P(l_k|r_i^w)$, have not been defined above. However, they can be obtained by applying Bayes' rule of conditional probabilities.

$$P(r_i^w|f_j) = \frac{P(f_j|r_i^w)P(r_i^w)}{P(f_j)} \tag{5.20}$$

$$P(r_i^w) = \sum_k P(l_k)P(r_i^w|l_k) \tag{5.21}$$

$$P(f_j) = \sum_i P(r_i^w)P(f_j|r_i^w) \tag{5.22}$$

Note that the probabilities $P(r_i^w)$ and $P(f_j)$ as defined above implicitly assume the traversal of only one edge. That is $P(r_i^w|l_k) = 0$ if $r_i^w$ and $l_k$ do not participate in the same edge, regardless of whether the node $r_i^w$ can be reached from $l_k$ by following a longer path. As such they only serve as a first approximation for the prior probabilities in Equation 5.19.

The next section discusses a method to acquire better estimates of the prior probabilities associated with reaching particular nodes in the graph by assuming the traversal of an arbitrarily large number of edges.

Reiterating the fact that the chain is reversible, as edges can be traversed in both directions, the probability of reaching a particular leaf node $l_k$ given the resource node $r_i^w$ is scaled by the probability of traversing the graph towards leaf nodes, rather than towards connected Wikipedia articles.

$$P(l_k|r_i^w) = P(leaf|r_i^w)P(l_k|r_i^w, leaf) \tag{5.23}$$

$$P(l_k|r_i^w, leaf) = \frac{P(r_i^w|l_k)P(l_k)}{P(r_i^w)} \tag{5.24}$$

The overall structure of the probabilistic adjacency matrix is shown in Figure 5.2.

### 5.2.1 Prior probability distribution

Once the probabilistic adjacency matrix has been compiled, it contains the probability of reaching a node in the graph from any other node, in one 'hop'. Now, by multiplying the matrix with its transpose one obtains the corresponding probability of reaching a node after the traversal of two edges. By repeating this process a large number of times we obtain:

$$\lim_{x \to \infty} M^x \tag{5.25}$$

,where $M$ stands for the adjacency matrix, the probability of reaching each node in the graph after traversing paths of arbitrary lengths. Thus, we obtain a prior probability distribution over all nodes in the graph. Using the spectral decomposition theorem, it can be shown that this is approximately equal to the leading eigenvector of the matrix.

$$\begin{array}{c|c|c|c}
 & \text{Leaves with experts, } l_k & \text{Domain resources, } r_i^w & \text{Connected Resources, } f_j \\
\hline
\text{Leaves with experts, } l_k & 0 & P(l_k|r_i^w) = \frac{P(leaf|r_i^w)P(r_i^w|l_k)P(l_k)}{P(r_i^w)} & 0 \\
\hline
\text{Domain resources, } r_i^w & P(r_i^w|l_k) & 0 & P(r_i^w|f_j) = \frac{P(f_j|r_i^w)P(r_i^w)}{P(f_j)} \\
\hline
\text{Connected Resources, } f_j & 0 & P(f_j|r_i^w) = P(link|r_i^w)P(f_j|r_i^w) & 0
\end{array}$$

FIGURE 5.2: The overall structure of the probabilistic adjacency matrix.

With respect to Equation 5.19, the elements of the leading eigenvector corresponding to resource nodes $r_i^w$ provide the prior probabilities, $P(r_i^w)$; furthermore, we use the average prior probability for the elements of $\varphi^*(\rho(u))$ as an estimate of the marginal probability, $P(\varphi^*(\rho(u)))$. Note that arbitrary definitions for the marginal probability can be used as it is ultimately a normalising constant.

A parallel can be drawn between the prior probabilities, $P(f_k)$, and the *random surfer* model used by Google's PageRank[13] extension to Kleinberg's *Hubs and Authorities* algorithm[44]. Similarly to the *random surfer*, each node in the graph is assigned a probability of being accessed without following a link. The difference however is that in this framework, each node is assigned a distinct prior probability, derived from the leading eigenvector of the probabilistic adjacency matrix.

| Profile used as input $\varphi^*(\rho(u))$ | Likelihood : $P(\varphi^*(\rho(u))\|r_i^w)$ | | Resources ranked in descending order of $P(r_i^w\|\varphi^*(\rho(u)))$ |
|---|---|---|---|
| | $\sum_j P(f_j\|r_i^w)$ | $\sum_j P(f_j\|r_i^w)\prod_k P(f_k)$ | |
| $\{f_1,f_2,f_3,f_4\}$<br>$\{f_1,f_2,f_3,f_5\}$<br>$\{f_1,f_2,f_3,f_6\}$ | $\{r_2^w,r_4^w,r_3^w,r_1^w\}$ | $\{r_2^w,r_4^w,r_3^w,r_1^w\}$ | |
| $\{f_2,f_3,f_4,f_5\}$<br>$\{f_2,f_3,f_4,f_6\}$<br>$\{f_2,f_3,f_5,f_6\}$<br>$\{f_1,f_2,f_4,f_5\}$<br>$\{f_1,f_2,f_4,f_6\}$<br>$\{f_1,f_2,f_5,f_6\}$ | | $\{r_4^w,r_2^w,r_3^w,r_1^w\}$ | |
| $\{f_1,f_3,f_4,f_5\}$<br>$\{f_1,f_3,f_4,f_6\}$<br>$\{f_1,f_3,f_5,f_6\}$ | $\{r_2^w,r_4^w,r_1^w,r_3^w\}$ | $\{r_4^w,r_2^w,r_1^w,r_3^w\}$ | |
| $\{f_1,f_4,f_5,f_6\}$<br>$\{f_3,f_4,f_5,f_6\}$ | $\{r_4^w,r_2^w,r_1^w,r_3^w\}$ | | |
| $\{f_2,f_4,f_5,f_6\}$ | $\{r_4^w,r_3^w,r_2^w,r_1^w\}$ | | |

TABLE 5.2: Recommendation lists produced using the two definitions of likelihood for the example of Figure 5.1, when profiles $\varphi^*(\rho(u))$ consist of four elements.

### 5.2.2 Likelihood

We provide the following definition for the likelihood, $P(\varphi^*(\rho(u))|r_i^w)$, as shown in Equation 5.19: It is the probability of reaching **any one** of the elements of $\varphi^*(\rho(u))$ that are connected to $r_i^w$ and **all** the remaining elements, starting from the node $r_i^w$. This is the case because we would like to introduce bias towards resources that are connected to as many elements of $\varphi^*(\rho(u))$ as possible. Formally:

$$P(\varphi^*(\rho(u))|r_i^w) = \sum_j P(f_j|r_i^w) \prod_k P(f_k) \tag{5.26}$$

such that $f_j \in \varphi^*(\rho(u)) \cap \varphi(r_i^w)$ , $f_k \in \varphi^*(\rho(u)) \backslash (\varphi^*(\rho(u)) \cap \varphi(r_i^w))$.

Initially, it was thought that the probability of reaching any of the of profiling elements from the node representing the resource considered for recommendation would be an adequate definition for the likelihood for this model. However, early experiments using this definition resulted in very poor quality recommendations. The recommendation lists produced for different inputs were found to be identical, and the elements at the top of the lists were dominated by resources with names that referred to places, or dates.

The cause of this failure, and the rationale behind the decision to effectively penalise resources for each profiling feature they are not connected to, can be illustrated using the example of Figure 5.1. Table 5.2 provides the recommendations produced for the example, when profiles $\varphi^*(\rho(u))$ consist of four elements.

| $l_1$ | $l_2$ | $r^w_1$ | $r^w_2$ | $r^w_3$ | $r^w_4$ | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.10 | 0.09 | 0.08 | 0.21 | 0.09 | 0.12 | 0.05 | 0.15 | 0.05 | 0.02 | 0.02 | 0.02 |

Table 5.3: The leading eigenvector of the probabilistic adjacency matrix constructed for the example given in Figure 5.1, normalised so that the sum of prior probabilities of all nodes is 1.

The aforementioned effect can be seen from the table, as using the early definition of likelihood (without the product over $f_k$) produces the same recommendation list for 9 out of the 15 possible profiles. Moreover, although resource $r^w_2$ is ranked at the top of the list, in six of these cases resource $r^w_4$ is connected to more profiling features than $r^w_2$. For this reason $r^w_4$ would intuitively be expected to be the most relevant resource to these profiles.

As traversals of edges from nodes that represent domain resources to nodes that represent other Wikipedia articles are equiprobable in our model,

$$P(f_2|r^w_4) = P(f_4|r^w_4) = P(f_5|r^w_4) = P(f_5|r^w_4) \tag{5.27}$$

and

$$P(f_1|r^w_2) = P(f_2|r^w_2) = P(f_3|r^w_2) \tag{5.28}$$

Thus, if a resource participates in a large enough number of edges the sum of the conditional probabilities of a number of nodes that appear in the input could be smaller than that assigned to a single node, given a resource which participates in a small number of edges. The effect present in our example however is a combination of two things: that $r^w_4$ participates in one more edge than $r^w_2$, but also

$$P(r^w_2) > P(r^w_4) \tag{5.29}$$

as $r^w_4$ is only connected to one of the two *mrkd*–tree leaves. This is also undesirable, as relevance to the input is judged to be more important for recommendation purposes than popularity. In any case, the revised definition causes the algorithm to correctly identify the six cases, and place $r^w_4$ at the top of the list by penalising $r^w_2$ for the three unconnected features.

To clarify, with respect to the example of Figure 5.1, consider the user profile:

$$\varphi^*(\rho(u)) = \{f_2, f_3, f_4, f_5\}$$

Now using the early definition of likelihood, $P(\varphi^*(\rho(u))|r_i^w) = \sum_j P(f_j|r_i^w)$:

$$\begin{aligned} P(\varphi^*(\rho(u))|r_2^w) &= P(f_2|r_2^w) + P(f_3|r_2^w) + P(f_4|r_2^w) + P(f_5|r_2^w) \\ &= 0.2 + 0.2 + 0 + 0 \\ &= 0.4 \end{aligned}$$

and

$$\begin{aligned} P(\varphi^*(\rho(u))|r_4^w) &= P(f_2|r_4^w) + P(f_3|r_4^w) + P(f_4|r_4^w) + P(f_5|r_4^w) \\ &= 0.2 + 0 + 0.2 + 0.2 \\ &= 0.6 \end{aligned}$$

$P(r_2^w|\varphi^*(\rho(u)))$ and $P(r_4^w|\varphi^*(\rho(u)))$ are given by:

$$\begin{aligned} P(r_2^w|\varphi^*(\rho(u))) &= \frac{P(\varphi^*(\rho(u))|r_2^w)P(r_2^w)}{P(\varphi^*(\rho(u)))} \\ &= \frac{0.4 \cdot 0.21}{P(\varphi^*(\rho(u)))} \\ &= \frac{0.084}{P(\varphi^*(\rho(u)))} \end{aligned} \qquad \begin{aligned} P(r_4^w|\varphi^*(\rho(u))) &= \frac{P(\varphi^*(\rho(u))|r_4^w)P(r_4^w)}{P(\varphi^*(\rho(u)))} \\ &= \frac{0.6 \cdot 0.12}{P(\varphi^*(\rho(u)))} \\ &= \frac{0.072}{P(\varphi^*(\rho(u)))} \end{aligned}$$

As $P(r_2^w|\varphi^*(\rho(u))) > P(r_4^w|\varphi^*(\rho(u)))$, resource $r_2$ is considered a more appropriate recommendation than $r_4$ by the framework. This is counter–intuitive, as $r_4$ is connected to more elements of the profile and should thus be judged more relevant.
However, with $P(\varphi^*(\rho(u))|r_i^w) = \sum_j P(f_j|r_i^w) \prod_k P(f_k)$ :

$$\begin{aligned} P(\varphi^*(\rho(u))|r_2^w) &= (P(f_2|r_2^w) + P(f_3|r_2^w))P(f_4)P(f_5) \\ &= (0.2 + 0.2) \cdot 0.02 \cdot 0.02 \\ &= 0.00016 \end{aligned}$$

and

$$\begin{aligned} P(\varphi^*(\rho(u))|r_4^w) &= (P(f_2|r_4^w) + P(f_4|r_4^w) + P(f_5|r_4^w))P(f_3) \\ &= (0.2 + 0.2 + 0.2) \cdot 0.05 \\ &= 0.03 \end{aligned}$$

The posterior probabilities obtained are thus:

$$P(r_2^w|\varphi^*(\rho(u))) = \frac{P(\varphi^*(\rho(u))|r_2^w)P(r_2^w)}{P(\varphi^*(\rho(u)))} \qquad P(r_4^w|\varphi^*(\rho(u))) = \frac{P(\varphi^*(\rho(u))|r_4^w)P(r_4^w)}{P(\varphi^*(\rho(u)))}$$

$$= \frac{0.00016 \cdot 0.21}{P(\varphi^*(\rho(u)))} \qquad = \frac{0.03 \cdot 0.12}{P(\varphi^*(\rho(u)))}$$

$$= \frac{0.00003}{P(\varphi^*(\rho(u)))} \qquad = \frac{0.036}{P(\varphi^*(\rho(u)))}$$

$$\therefore \qquad P(r_4^w|\varphi^*(\rho(u))) > P(r_2^w|\varphi^*(\rho(u)))$$

It is noted that the same behaviour is observed using either definition when three profiling features are connected to $r_2^w$. The same statement applies to $r_4^w$, when the profiles do not include $f_2$. In addition, $f_2$ is connected to all four resources, as shown in Figure 5.1. Such nodes also appear in real situations. For example, the Wikipedia article about any motion picture filmed in Hollywood would contain links to the articles 'USA' and 'Hollywood'. Similarly, in the music domain Wikipedia articles about recording artists will contain links to articles about the year each recording was released.

Thus, the application of the early definition in a real situation yields recommendations whose corresponding Wikipedia articles contain a relatively small number of links. Moreover, the framework is driven by the latent information carried in hyperlinks. For this reason, the recommendations produced in this way are effectively the resources for which the least amount of information is available.

Having provided a definition for all the elements of

$$P(r_i^w|\varphi^*(\rho(u))) = \frac{P(\varphi^*(\rho(u))|r_i^w)P(r_i^w)}{P(\varphi^*(\rho(u)))} \tag{5.30}$$

the following section provides the results from experiments carried out to assess recommendation lists produced by the framework.

## 5.3 Evaluation

To evaluate the various aspects of the framework, the three datasets introduced in Section 4.3 have been processed as follows:

Last.fm: 19 667 of the 141 078 artists available in Last.fm were mapped to Wikipedia pages about them via the Musicbrainz KB. Subsequently, 857 492 other Wikipedia articles were found to be connected to these via hyperlinks. The 810 104 Last.fm users were first sampled (as in Section 5.1.1) to obtain a subset of 50 000. In Last.fm, the number of times each artist has been played is recorded instead of an explicit rating. While this does not affect the fact the the value is proportional to the level of preference relative to other artists, it does mean that the normalised ratings (see Section 5.1.2) will vary more between users depending on how much they use the system. Thus a larger threshold was set for the widest dimension of the *mrkd–tree* used to summarise the sample, in order to account for the larger variance. The threshold was set empirically to 2, producing 35 282 leaf nodes. Therefore, the recommendation space for Last.fm, $S_L$, is a graph containing 912 441 nodes and 4 529 942 edges. The leading eigenvector of the probabilistic adjacency matrix which encodes $S_L$ was computed by carrying out the *Arnoldi iteration* [4] 50 times.

Netflix: Wikipedia pages have been identified for 10 088 of the 17 770 movies, which in turn were connected to a further 929 381 pages. A *mrkd–tree* was constructed for a sample of 150 000 NetFlix subscribers from the total of 2 649 429. The tree was built by setting the widest dimension to 1, to yield 26 886 leaf nodes. As such, the recommendation space spanning the NetFlix domain is a graph of 966 355 nodes and 13 954 416 edges. As above the leading eigenvector was approximated with 50 applications of the *Arnoldi iteration.*

del.icio.us: 1 965 users from the Last.fm dataset were identified by Szomszor et. al., [85], to hold del.icio.us subscriptions. As such, this dataset provides a unique opportunity to evaluate the framework's performance in producing cross–domain recommendations using labeled data. The users bookmarked a total of 598 660 URLs, described using 103 339 distinct tags. The tags were then processed as in Section 4.1, resulting in identifying 27 027 Wikipedia articles to which tags could be mapped to without requiring disambiguation. These were found to be connected to a further 2 349 685 articles. The sampling and summarisation steps have not been carried out for this dataset, as the number of available experts is sufficiently small. As such, a recommendation space $S_D$ was constructed for del.ico.us with 2 378 677 nodes, and 11 689 168 edges, and encoded in a probabilistic adjacency matrix. The leading eigenvector is obtained using the same process as above.

The following section sets out to assess the framework's capabilities in producing single domain recommendations, using the datasets presented above. In the light of the observations made, Section 5.3.2 provides an assessment of qualitative changes in system behaviour when producing cross–domain recommendations.

### 5.3.1 Assessing the quality of the recommendations produced

To carry out this aspect of evaluation, each dataset was sampled to obtain a set of 100 experts, as per Section 5.1.1. Then, one resource was removed from each profile, and kept for testing. The remainder of the profiles is then used to produce recommendations from the same domain, which are subsequently evaluated using the following metrics:

Popularity: The number of ratings a resource has received, normalised by the number of ratings assigned to the most popular resource.

$$Popularity(r_i) = \frac{|\varepsilon(r_i)|}{\underset{r_j \in \rho^*(d)}{\arg\max}(|\varepsilon(r_j)|)} \tag{5.31}$$

The popularity of a list of recommendations is given by the mean popularity of the resources contained within it.

Ratability: Ratability measures how likely each resource is to be rated next by a user, and coincides with the definition of posterior probabilities assigned to resources, given the user input.

$$Ratability(r_i^w) = \log(P(r_i^w | \varphi^*(\rho(u)))) \tag{5.32}$$

Again, to obtain an overall score for a recommendation list, ratability is averaged over all resources in the list.

Intra–list similarity between rating vectors (ILSM–RV): The cosine vector similarity between rating vectors:

$$\vec{\nu}_{r_i} = \langle \nu(e_1, r_i), \nu(e_2, r_i), \cdots, \nu(e_{|\varepsilon^*(d)|}, r_i) \rangle \tag{5.33}$$

Then

$$\text{ILSM–RV}(r_i, r_j) = \frac{\vec{\nu}_{r_i} \cdot \vec{\nu}_{r_j}}{\|\vec{\nu}_{r_i}\| \; \|\vec{\nu}_{r_j}\|} \tag{5.34}$$

The ILSM–RV score for a recommendation list is given by the average similarity between all pairs of resources in the list.

Intra–list similarity between recommendation space vectors (ILSM–WV): The similarity of rows in the adjacency matrix encoding the domain's recommendation space, that correspond to the recommended resources.

$$\text{ILSM–WV}(r_i, r_j) = \frac{\vec{\varphi}(r_i) \cdot \vec{\varphi}(r_j)}{\|\vec{\varphi}(r_i)\| \|\vec{\varphi}(r_j)\|} \tag{5.35}$$

The ILSM WV score for a recommendation list is given by the average similarity between all pairs of resources in the list, as above.

Authority: As explained in Section 5.2.1, the prior probabilities assigned to resources express the probability of landing on each node in the recommendation space, after having traversed an arbitrarily large number of edges. Thus, these probabilities express the centrality of each resource in the recommendation space, providing an intuitive definition for the authority metric. In order to facilitate the comparison of authority scores for different domains, the authority score is normalised using the prior probability assigned to the most authoritative resource.

$$Authority(r_i) = \log(P(r_i^w)) - \log\left(\arg\max_{r_j \in \rho^*(d)} (P(r_i^w))\right) \tag{5.36}$$

The authority score for a recommendation list is then the average authority of the resources contained within it.

Rank: The rank of the withheld resource in the list of recommendations produced based on the corresponding profile.

Personalisation: The personalisation score for two lists of recommendations of size $l-1$, $rec_d^l(e_i)$ and $rec_d^l(e_j)$ is given by Spearman's Footrule [82]:

$$F^l(rec_d^l(e_i), rec_d^l(e_j)) = \sum_{r_k \in rec_d^l(e_i) \cup rec_d^l(e_j)} |rank(r_k, rec_d^l(e_i)) - rank(r_k, rec_d^l(e_j))|, \tag{5.37}$$

where $rank(r_k, \sigma_j)$ is the rank of resource $k$ in list $\sigma_j$.

For $r_i \notin rec_d^l(e_j)$, $rank(r_i, rec_d^l(e_j))$ returns the constant, $l$, typically set to 1 larger than the size of the lists. The score measures the total difference in ranks for all the resources contained in both lists, while resources that do not appear in the list are given rank $l$. The overall personalisation score assigned to each list is the average personalisation between the list in question, and all other lists produced in the experiment.

With the exception of the rank metric, only the 50 resources with the largest posteriors assigned to them are considered in calculating the scores presented above. Table 5.4 presents the mean scores obtained for each dataset, giving an overview of the qualitative characteristics of the recommendation lists produced by the framework.

A first observation is that the ratability score varies greatly across the three domains. In addition, recommendations from the Netflix domain appear to be more precise than their Last.fm counterparts, as reflected by the average rank scores. This, in contrast to the fact the Last.fm recommendations receive much higher posteriors than Netflix can give a basis to the argument that the absolute value of the posterior probability assigned to a resource based on a profile cannot be used as the single means to assess how appropriate

| | Netflix | Last.fm | del.icio.us |
|---|---|---|---|
| Popularity | 0.011 | 0.012 | 0.252 |
| Ratability | -1 010.77 | -279.14 | -4 370.45 |
| ILSM RV | 0.0474 | 0.0006 | 0.2103 |
| ILSM WV | 0.043 | 0.307 | 0.033 |
| Authority | 0.25 | 0.29 | 0.15 |
| Rank | 675 | 710 | 2 247 |
| Personalisation | 0.48 | 0.31 | 0.66 |

Table 5.4: Mean metric scores for recommendation lists produced for the domains Netflix, Last.fm, and del.icio.us.

the recommendations will be. While personalisation scores are relatively high for all domains, the fact that the recommendation lists appear to be more personalised for Netflix subscribers than for Last.fm ones gives another indication that the magnitude of the posterior probability is not proportional to the quality of a recommendation. This is not necessarily a negative result however, provided that resources are still ranked in the correct order.

Secondly, by comparing the scores of the two ILSM metrics to the the analysis performed in Section 4.3, it can be seen that the relationships between the two representations identified there are still present. That is to say the similarity between the Wikipedia projections of resources, given by the ILSM WV metric, fall in the range expected based on the similarity of their corresponding rating vectors.

Figure 5.3 presents a detailed view of the authority and popularity scores assigned to each list of recommendations, in each domain. The strong correlation shown means that authoritative resources are also popular, and vice versa. This is interpreted as evidence that the probabilistic model defined over the recommendation space captures the intrinsic structure of the dataset very well, in all three domains.

Regardless of this encouraging result, the rank of the withheld resources in the recommendation lists produced is very low, on average. In order to examine the predictive accuracy of the framework in more detail, Figure 5.4, provides a histogram of the distribution of ranks for the withheld resource in each domain. In addition, the second column of graphs gives the *Coverage* in each domain, as a function of the size of recommendation lists provided. This is the fraction of withheld resources retrieved, and is given by the cumulative distribution function (CDF) of the rank distribution.

It can be seen from the figure that in all domains, the withheld resources appear in the first bin (top 75 recommendations) more frequently than any other interval of the same size. This is reflected in the coverage graphs by the steep gradient of the curves when the size of recommendations is small.

The recommendations produced for Netflix seem to be the most accurate of the three domains, as demonstrated by the shape of Figure 5.4(a). This indicates that the withheld

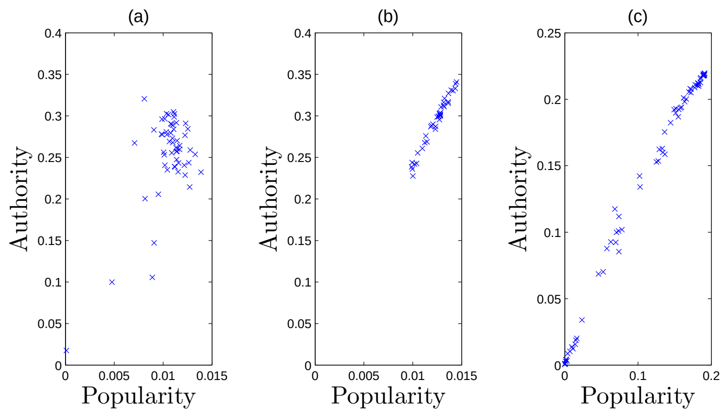


FIGURE 5.3: The mean authority shown against the corresponding mean popularity for each recommendation list in the domains: (a) Netflix, (b) Last.fm, and (c) del.icio.us.

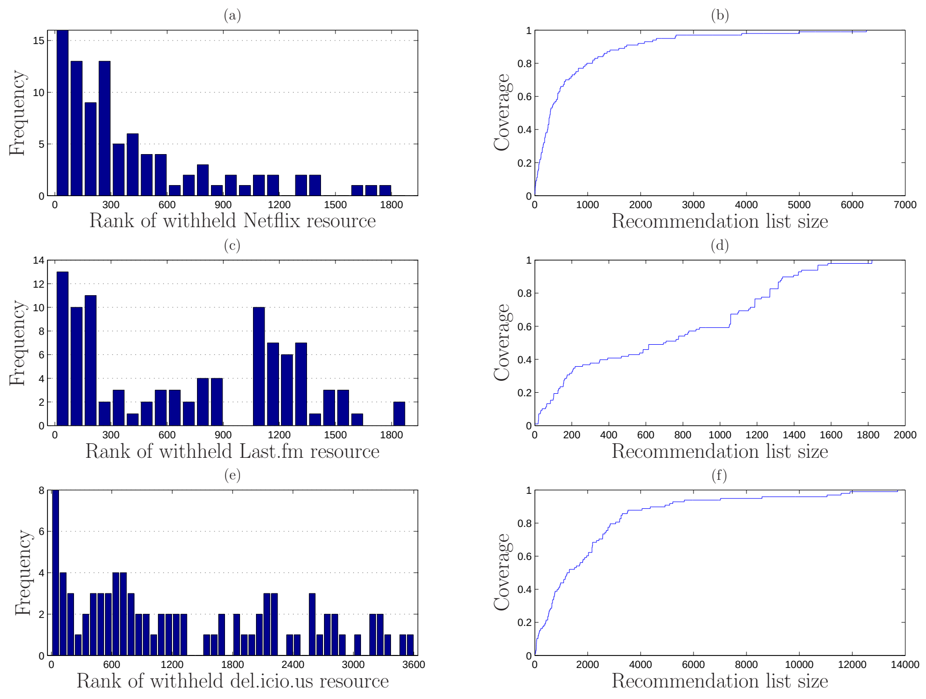


FIGURE 5.4: (a) Histogram of the rank distribution for the withheld Netflix resources in the corresponding recommendation lists. Each bin contains 75 ranks.
(b) Coverage as a function of the number of resources in each recommendation list for the Netflix domain.
(c)–(d) Corresponding figures for the Last.fm domain.
(e)–(f) Corresponding figures for the del.icio.us domain.

resource becomes less likely to appear low in the recommendation list and is confirmed by the shape of the corresponding CDF in Figure 5.4(b).

Moreover, the coverage curve for Netflix is steeper than both other domains. In comparison, by recommending the top 200 resources retrieves approximately 40% of the withheld resources, both in Last.fm and Netflix, while in del.icio.us recommendation lists need to consist of over 1 000 resources to achieve 40% coverage. However, 60% of the withheld Netflix resources are retrieved by increasing the size of the lists to 400, while the same increase in Last.fm incurs negligible changes in coverage.

**Profile coherence**

While for both Last.fm, and del.icio.us (in figures 5.4(c) and (e) respectively) the first bin of the histogram is the most frequent one, the withheld resources receive specific low ranks with high frequency. In order to explain this observation an assessment of the characteristics of the profiles used, and the corresponding withheld resources was carried out. The profiles are split based on the rank of the withheld resource in the corresponding lists of recommendations. Each subset consists of 75 consecutive ranks, corresponding to the bin size of the histograms in Figure 5.4.

Then, for each profile the vector cosine similarity score is calculated between the withheld resource and each of the remaining resources, and averaged to obtain a similarity score between the withheld resource and the entire profile. The average score for all profiles in each bin can then indicate whether the similarity of the withheld resource to the remainder of the profile is related to its ranking in the recommendation list produced.

In addition, the similarity between the resources contained in each profile is calculated using rating vectors, and used to assess whether profiles which are more coherent (in that they contain similar resources) give rise to recommendation lists where the withheld resource is ranked near the top.

Figure 5.5 presents the results of the analysis described above, for the Last.fm domain. Figure 5.4(c) is reproduced in Figure 5.5(a) to ease visual comparisons. Figure 5.5(c) reveals that the resources contained in the profiles for which the framework performs badly, are relatively very similar to each other. The withheld resource however is not. The interpretation given is that the withheld resource appears to be unrelated to the rest of the profile. Thus, it is ranked low down the list and assigned a rank effectively at random, demonstrated by the shape of the distribution. This gross observation has been confirmed by visually inspecting a subset of the profiles in question.

Moreover, a second distinct case where the framework performs badly can be identified. For the single case that the withheld resource is very similar to other resources in the profile, but receives a very low rank, we find that the profile itself consists of very dissimilar resources. It is thus argued that in this case, no coherent pattern is defined

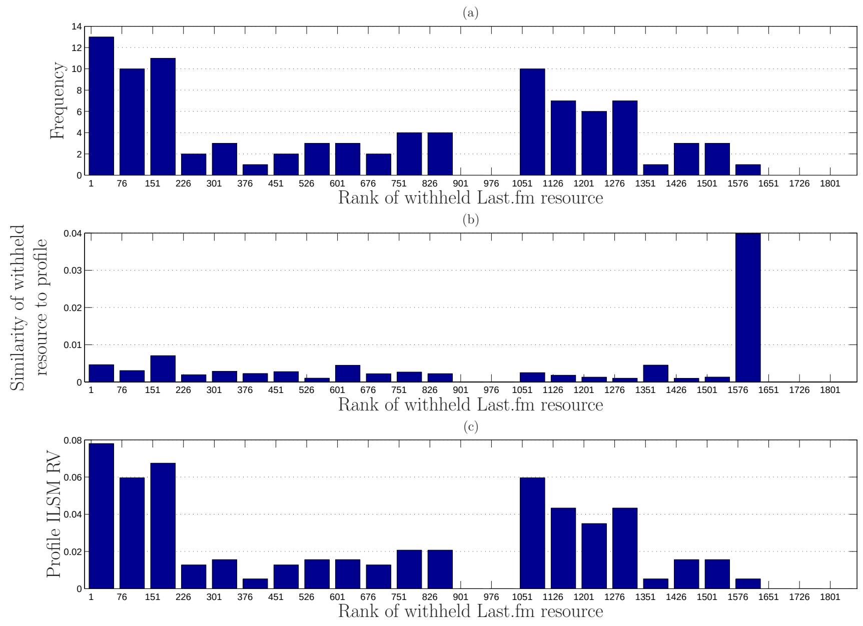


FIGURE 5.5: (a) Histogram of the rank distribution for the withheld Last.fm resources in the corresponding recommendation lists. Each bin contains 75 ranks.
(b) The similarity of the withheld resource to the remainder of the profile, averaged over the cases in each bin of (a).
(c) The ILSM RV score of the remaining profile resources, averaged over the cases in each bin of (a).

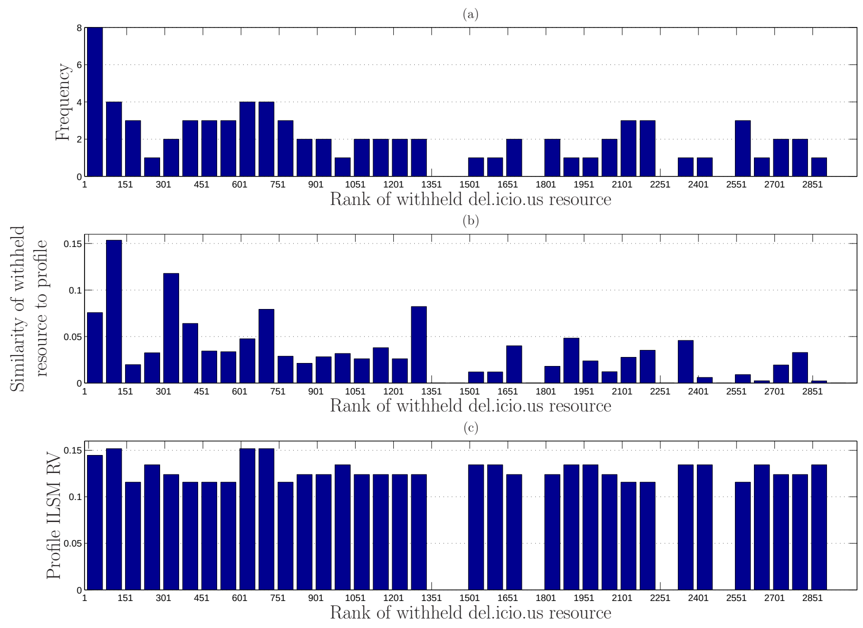


FIGURE 5.6: (a) Histogram of the rank distribution for the withheld del.icio.us resources in the corresponding recommendation lists. Each bin contains 75 ranks.
(b) The similarity of the withheld resource to the remainder of the profile, averaged over the cases in each bin of (a).
(c) The ILSM RV score of the remaining profile resources, averaged over the cases in each bin of (a).

| | Adapt–Light | Adapt–Even Split | Adapt–Heavy |
|---|---|---|---|
| Netflix | 0.22 | 0.17 | 0.11 |
| Last.fm | 0.17 | 0.14 | 0.08 |
| del.icio.us | 0.44 | 0.35 | 0.24 |

TABLE 5.5: Mean adaptability scores for three sets of 50 pseudo–users in each domain.

by the profile for the machine learning algorithm to identify, and as such the withheld resource is ranked very low in the recommendation list.

Figure 5.6 provides the corresponding analysis carried out for the del.icio.us domain. As shown by Figure 5.6(c), the profiles are equally similar throughout the distribution of ranks. However, the similarity of the withheld resource to the remaining ones diminishes as it appears lower down the list. This observation supports the claims made earlier, with respect to the Last.fm domain: the framework fails to give a high rank to the withheld resource if it is very dissimilar to the rest of the profile. Moreover, it can be argued that as the withheld resource becomes less and less similar to the rest of the profile, it thus ranked lower down the list but still in a random fashion.

**Adaptability**

In addition to the types of analysis presented above, an assessment of how well changes in profiles propagate through the algorithm to affect the ranking of recommendations has been carried out. To do so, each dataset was randomly sampled to obtain 50 distinct pairs of experts. The profile of each expert was then split into four segments of equal size, and three pseudo–users constructed from each pair:

Adapt–Light: Consists of the first three segments from the first expert's profile, and the first segment of the second user's profile.

Adapt–Even Split: Contains resources in the first two segments in both profiles.

Adapt–Heavy: The union of the first segment of the first expert's profile and the first three segments of the second users profile.

Each pseudo–user is then used as input, and the top 50 recommendations recorded. The rankings are then compared with the last segment of the second profile, which has not been used to create pseudo–users. The comparison is carried out using Spearman's Footrule, normalised by the upper bound obtained when the two lists have no elements in common. The results are presented in Table 5.5.

It can be seen from the table that as more elements from the second profile are included in the pseudo–users, the more similar the recommendation lists become to the unused segment of the same profile. Thus, it appears that the framework is successful in adapting recommendation lists in a meaningful way, in response to corresponding changes in user profiles.

### 5.3.2 Assessing the feasibility of producing cross–domain recommendations

Having established the ability of the proposed framework to produce meaningful recommendations within a single domain, this section provides a comparative evaluation of its performance in producing cross-domain recommendations. Four experiments have been designed to carry out this aspect of evaluation, exploiting the fact that 1 965 Last.fm subscribers also hold del.icio.us accounts. It is worth noting that the size of this overlap is very small compared to the total number of subscribers for these two systems. Thus, a selection effect is present, in that only del.icio.us users with an expressed interest in music (through their Last.fm account) are considered. While it has been verified by visually inspecting the dataset that neither the URLs bookmarked nor the tags used are restricted to the music domain, the conditions for selecting an unbiased sample from del.icio.us (with respect to the subscriber's domains of interest) are unclear. As such speculations on the frameworks' performance with respect to del.icio.us users with different (or unbiased) characteristics cannot be made. Moreover, the methodology used in carrying out the experiments presented in this section relies on the identification of a common set of experts in at least two domains.

Each experiment is carried out as follows, using 100 randomly selected users who appear in both domains:

**Experiment A:** URLs bookmarked by the selected users in del.icio.us are used as input to the algorithm, to produce recommendations from the Last.fm domain. The lists of recommendations are then compared against the available Last.fm profiles for the same users.

**Experiment B:** This is the inverse scenario to the previous experiment. The 50 artists most frequently listened to by the users selected for evaluation are used as input, to generate recommendations from the del.icio.us domain. The URLs that the users have actually bookmarked in del.icio.us are used to validate the recommendations produced.

**Experiment C:** The ability of the system to produce cross–domain recommendations can be demonstrated by showing that the recommended resources carry the same information regarding user interests as the original profiles, albeit in different domains. To do so, the del.icio.us profiles of users selected for this experiment are used to recommend movies for the Netflix domain. The top 50 recommendations are then kept, and fed back into the algorithm to produce Last.fm recommendations. Comparing the results of this experiment to those obtained in Experiment A can then reveal whether replacing the original del.icio.us profiles with the recommendations produced for a third domain has a detrimental effect on the quality of recommendation lists in the Last.fm domain.

| Experiment | **A** | **B** | **C** | **D** |
|---|---|---|---|---|
| Profiling features domain<br>Intermediate domain<br>Recommendation domain | del.icio.us<br>–<br>Last.fm | Last.fm<br>–<br>del.icio.us | del.icio.us<br>Netflix<br>Last.fm | Last.fm<br>Netflix<br>del.icio.us |
| Popularity | 0.15 | 0.0758 | 0.1169 | 0.017 |
| Ratability | -60 946 | -128 619 | -499 157 | -3 308 447 |
| ILSM RV | 0.004 | 0.0273 | 0.0001 | 0.022 |
| ILSM WV | 0.1056 | 0.2549 | 0.1075 | 0.1666 |
| Authority | 0.12 | 0.05 | 0.0905 | 0.0061 |
| Profile rank | 2 911 | 9 369 | 3 138 | 1 918 |
| Top rank | 42 | 721 | 47 | 508 |
| Personalisation | 0.33 | 0.2025 | 0.2308 | 0.1569 |

TABLE 5.6: Mean metric scores for recommendation lists produced for the four cross–domain recommendation experiments.

**Experiment D:** This experiment shares the same intuition as the previous one, using Netflix as an intermediate domain between del.icio.us and Last.fm. Here however the process is carried out in the reverse order. The top 50 recommended movies based on the available Last.fm profiles are first obtained, and in turn used as input to recommend URLs found in the del.icio.us domain. The system's performance can then be evaluated based on the del.icio.us profiles available for the same users, and compared with the results of Experiment B.

Table 5.6 gives the average score of the recommendation lists produced in each experiment, using the metrics introduced in the previous section. However, as the number of withheld resources for each user is equal to the size of their profile in the recommendation domain, two rank metrics can be defined. Thus, the 'Profile rank' metric gives the average rank of all withheld resources in the corresponding recommendation list. In contrast the 'Top rank' for each profile is the rank of the withheld resource ranked highest in the corresponding list.

It is noted once more that the points defined by the scores of the ILSM metrics fall on the curves showed in figures 4.1(c) and 4.3(c) when the recommendation domain is Last.fm or del.icio.us respectively, confirming the quality of the Wikipedia projections. Moreover, while the average values for the Popularity and Authority scores reported in Table 5.6 can be seen to contradict the correlation identified in Figure 5.3, this is simply an artifact of of the averaging process, as it is very sensitive to extreme values. The correlation is easily retrieved by performing the same type of analysis as before, by plotting the Popularity and Authority scores of all individual recommendations. However, it is omitted from this section, as it would amount to nothing more than carrying out the same analysis using a different set of resources from the same spaces.

Comparing the rank scores in Table 5.4 to those obtained here, by considering the rank of all resources that appear in the withheld profiles indicates that the system performs better when the recommended resources appear in the same domain as the elements of user profiles. However, the ‘Top rank’ scores across all four experiments are much lower. Thus, at least one resource from each profile can be expected to be ranked much closer to the top of the list than most others. The reader is reminded of the discussion provided in the previous section on the negative effects of randomly selecting a single resource from each profile to evaluate the predictive accuracy of a system. It was argued that if the withheld resource is not similar to the others referenced by the same profile, the system cannot be expected to assign a high posterior probability to it. By averaging the ‘Profile rank’ score, all these cases will have a high influence on value obtained. The reverse effect is present in the ‘Top rank’ metric, as the selection of resources is now effectively biased towards those the system performs best. Thus an objective comparison of these scores cannot be carried out directly in order to claim that cross–domain recommendations are of higher quality than their single domain counterparts. The results presented so far however, do indicate that in some cases cross–domain recommendations can be very accurate using the framework.

A number of interesting observations can be made by comparing the two rank metrics over the four experiments. When the recommendation domain is Last.fm (experiments A and C) the system suffers very small losses in predictive accuracy (as captured by the two rank metrics) by replacing the original del.icio.us profiles with the top 50 recommended resources from the Netflix domain. More impressively, the accuracy of del.icio.us recommendations actually improves when Netflix is used as an intermediate domain (Experiment D), in comparison to Experiment B, where del.icio.us recommendations are produced directly based on the original Last.fm profiles. In addition, the ‘Profile rank’ score obtained for Experiment D is the only case where it is lower than the corresponding del.icio.us score in Table 5.4. In other words all the withheld del.icio.us resources appear closer to the top of the list on average, by considering resources recommended by the system as input, rather than the actual user–generated data. This is a situation where it is clearly beneficial for a RS to consider multiple recommendation domains, as it improves the system’s predictive accuracy.

Moreover, the personalisation level for Last.fm recommendations is comparable when a single, two, or all available evaluation domains are used. However, the personalisation score for lists of del.icio.us recommendations drops from the extremely high 0.66 score for single domain recommendations, to 0.15 when the recommendations are most accurate in Experiment D. Thus, very large personalisation scores could be interpreted to indicate that the system has failed to identify similar patterns in user profiles, resulting in very varied recommendation lists. As members of social networks are typically expected to share common interests, such an observation could identify a lack of sufficient information to express user interests in that domain. If that is the case, by using resources in a different domain, judged relevant to the available profiles by the system, additional

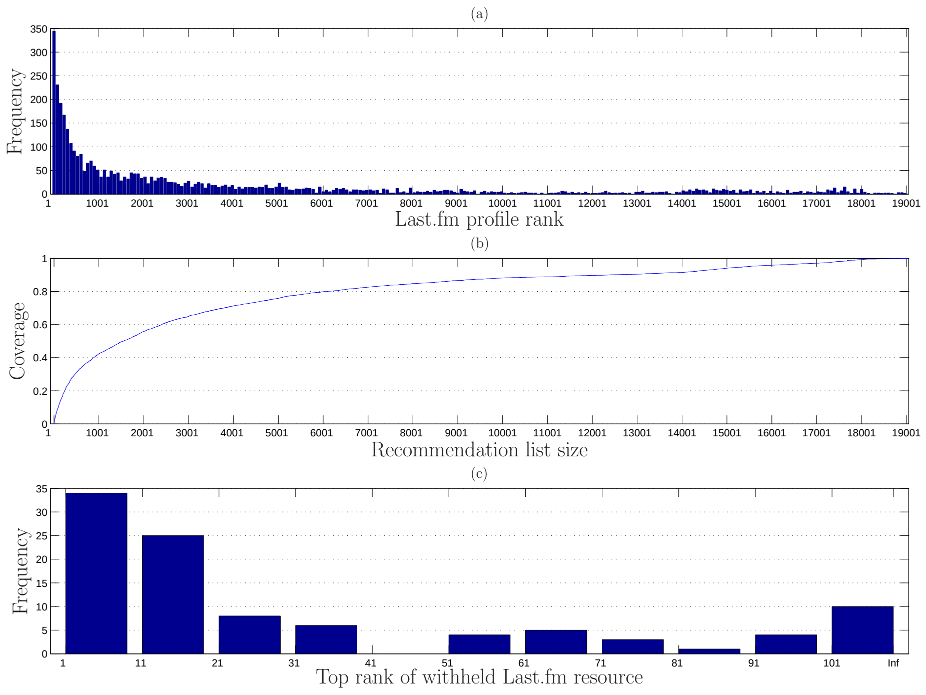


FIGURE 5.7: **Experiment A:**
(a) Histogram of the rank distribution for all the withheld Last.fm resources in the corresponding recommendation lists. Each bin contains 75 ranks.
(b) Coverage as a function of the number of resources in each recommendation list for the Last.fm domain.
(c) Histogram of the rank distribution for the withheld Last.fm resources that are ranked the highest in the corresponding recommendation lists (1 for each Last.fm expert). Each bin contains 10 ranks

information about user interests may be obtained, which can further be exploited to obtain more accurate recommendations.

With that in mind, it is observed that the posteriors assigned to resources here are much lower than those reported in Table 5.4. This adds support to the argument that resources that are ranked highly would have the same positive qualities, regardless of the absolute value of the posterior probability assigned to them.

Figures 5.7 and 5.8 provide a more detailed analysis of the rank distribution for the withheld resources in Experiments A and C, respectively. Each figure consists of three graphs. The top graph is a histogram of the rank distribution for all the Last.fm resources that appear in the 100 profiles selected for evaluation. The second one provides the coverage of the same set of resources as a function of the number of recommendations provided each time. Finally, the bottom graph of each figure provides a histogram of the 'Top rank' distribution – the lowest rank a withheld resource is assigned per profile.

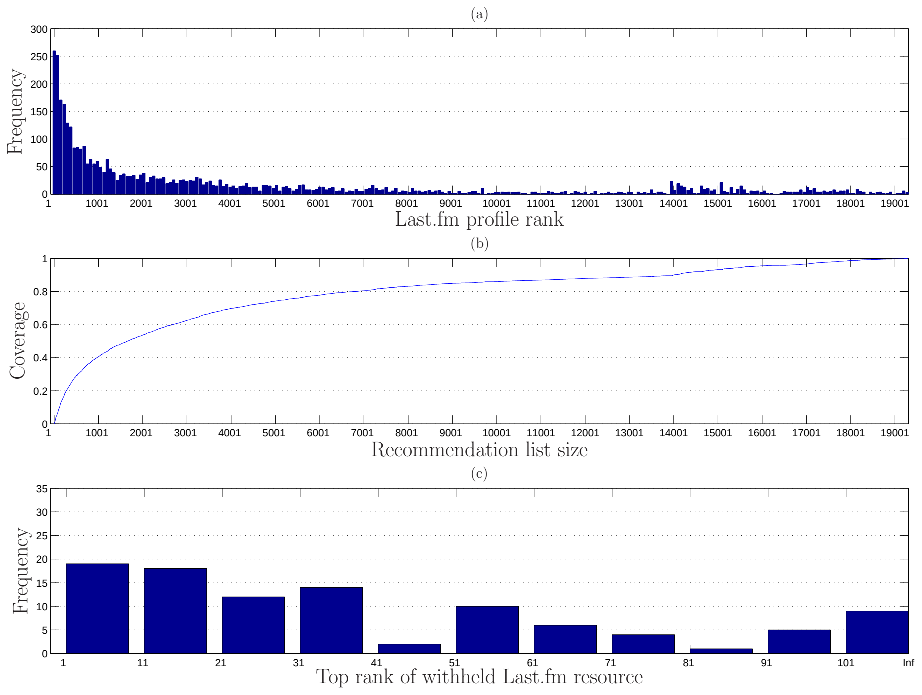


FIGURE 5.8: **Experiment C:**
(a) Histogram of the rank distribution for all the withheld Last.fm resources in the corresponding recommendation lists. Each bin contains 75 ranks.
(b) Coverage as a function of the number of resources in each recommendation list for the Last.fm domain.
(c) Histogram of the rank distribution for the withheld Last.fm resources that are ranked the highest in the corresponding recommendation lists (1 for each Last.fm expert). Each bin contains 10 ranks

It is interesting to note that the top two graphs in both figures are extremely similar. This adds further evidence to the claim that when the recommendation domain is Last.fm, the utility of the user created del.icio.us profiles to produce recommendations is equivalent to that of recommended Netflix resources based on the same profiles. Thus, the Netflix recommendations must be accurate, as they are found to convey the same latent information about user interests. A comparison of figures 5.7(b) and 5.8(b) to Figure 5.4(f) would appear to indicate that performance severely degrades in producing cross domain recommendations as 40% coverage is only achieved when approximately 1 000 recommendations are provided each time (as opposed to 200 reported earlier). We re–iterate however that a direct comparison between these scores cannot be objective, as different selection effects are present.

A close inspection of Figure 5.7(c) reveals that for an extremely high 59 of 100 cases, using a person's del.icio.us profile to recommend Last.fm artists results in at least one true positive in the top 20 recommendations. In addition, by recommending the top 100

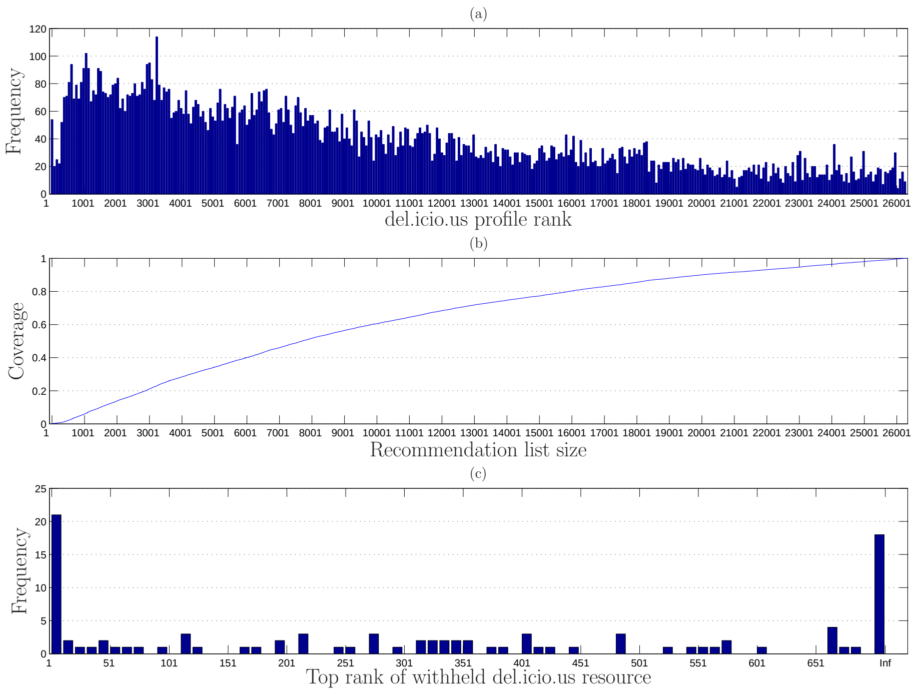


FIGURE 5.9: **Experiment B:**
(a) Histogram of the rank distribution for all the withheld del.icio.us resources in the corresponding recommendation lists. Each bin contains 75 ranks.
(b) Coverage as a function of the number of resources in each recommendation list for the del.icio.us domain.
(c) Histogram of the rank distribution for the withheld del.icio.us resources that are ranked the highest in the corresponding recommendation lists (1 for each del.icio.us expert). Each bin contains 10 ranks

resources only 10 lists are found to not contain any resources actually listened to by the same person. In comparison Figure 5.8(c) shows that in fact the system performs slightly worse when recommendations from Netflix are used instead of the original del.icio.us profiles, as withheld resources appear in the top 20 recommendations for only 37 cases. However, by recommending the top 40 resources at least one true positive is retrieved for 63 cases.

Figures 5.9 and 5.10 provide the corresponding analysis for the rank distribution of the withheld resources in experiments B and D, when resources from the del.icio.us domain are recommended. The system performs clearly worse here than before. As Figure 5.9(a) shows, for the first time withheld resources do not most frequently appear at the top of the list. This observation is consistent with the findings of the previous section, where recommendations from the del.icio.us domain were found to be of the lowest quality. However, Figure 5.10(a) shows that the system's performance improves when the Last.fm profiles provided by users are replaced by recommendations from the

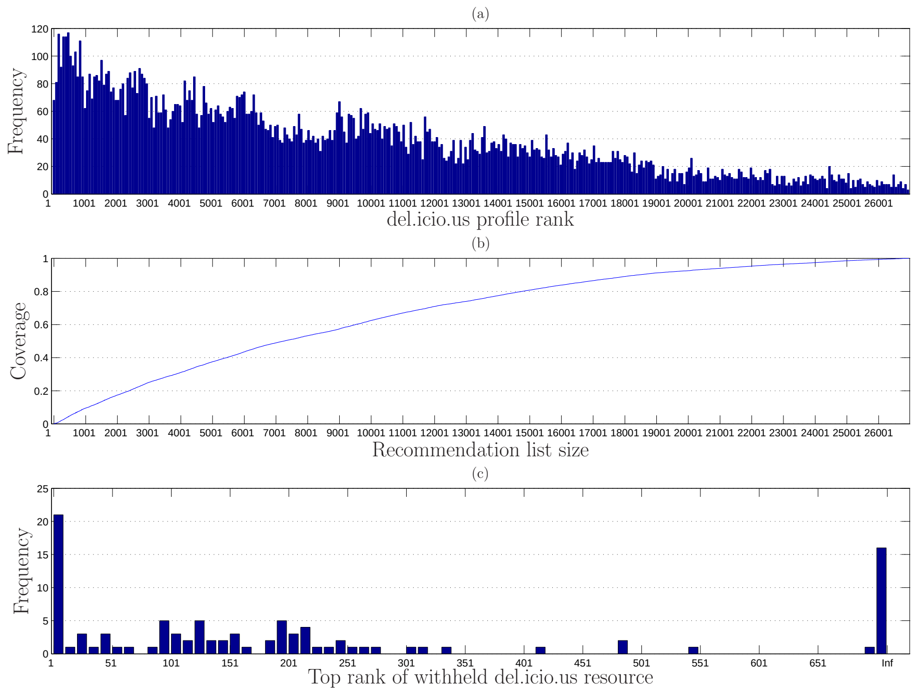


FIGURE 5.10: **Experiment D:**
(a) Histogram of the rank distribution for all the withheld del.icio.us resources in the corresponding recommendation lists. Each bin contains 75 ranks.
(b) Coverage as a function of the number of resources in each recommendation list for the del.icio.us domain.
(c) Histogram of the rank distribution for the withheld del.icio.us resources that are ranked the highest in the corresponding recommendation lists (1 for each del.icio.us expert). Each bin contains 10 ranks

Netflix domain. This is the case as many more withheld resources appear closer to the top of the final recommendation lists produced.

In both cases, coverage is consistently lower in these experiments than when recommendations are made for resources in the Last.fm domain, as shown in figures 5.9(b) and 5.10(b). This is in agreement with the system's performance in producing single–domain recommendations for the del.icio.us domain.

The distribution of the 'Top rank' metric for these experiments, presented in figures 5.9(c) and 5.10(c) provides a somewhat more positive view of the framework's performance in this domain. In both cases the withheld resource with the highest rank for each profile appears in the top 10 recommendations for the same profile much more frequently than any other rank interval of the same size. However, recommendations for more than 700 resources must first be made before a true positive is retrieved for a similarly large portion of user profiles.

This section has provided an analysis of the results obtained by carrying out four cross–domain recommendation experiments. This has provided evidence that the automated recommendation of resources from domains different to that of profiling elements is not only feasible, but can also be accurate and in some cases beneficial:

- Recommendations for Last.fm resources, based on del.icio.us profiles, were found to be more accurate than recommendations for del.icio.us resources based on the same profiles.
- Accurate recommendations for Netflix resources can be made using either Last.fm or del.icio.us profiles. The system's behaviour is almost identical in recommending Last.fm resources based on corresponding del.icio.us profiles, or based on Netflix resources recommended previously for the same del.icio.us users. Moreover, the predictive accuracy of the algorithm actually improves when Last.fm profiles are replaced with Netflix resources in the same way, in order to recommend del.icio.us resources.

## 5.4 User Profiling: the Semantic Logger

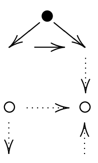


Fig. 1.2

The Semantic Logger (SL) is an autobiographical metadata acquisition system, designed to unobtrusively capture any information created or accessed by users. It is the outcome of previous joint work, published in [89, 23] and described previously in Section 2.4

This section demonstrates how profiles generated by integrating information captured by the SL system can be used to produce recommendations through the proposed framework. It mainly serves to illustrate the fact that recommendations can be produced based on information substantially different to the type of recommended resources (or their domain of origin).

### 5.4.1 Obtaining mappings between captured information and Wikipedia

Once information has been captured and stored, it must then be mapped to Wikipedia, the universal vocabulary, in order to be used in producing recommendations. This section provides details on how this is carried out for the various types of information.

#### 5.4.1.1 Local information

Email messages

Only emails sent directly to, or composed by the user are considered. This choice was made since the information contained in other messages could be substantially less relevant to the user. GATE [17] was used to process the body of each message to obtain concepts annotated as *Organization*, *Person*, or *Address*. Wikipedia is queried directly for instances of the former two. If a page is returned with 100% relevance, it is included in the user's profile. Concepts annotated as *Address* are filtered, and only resolvable URLs are kept. The social bookmarking site, del.icio.us [93], exposes descriptive keywords assigned by its' users to the URLs they bookmark. If such tags can be obtained for the URLs identified by GATE, they are matched to Wikipedia entries through the process identified in Section 4.1. Any such entries are added to the user's profile, $\rho(u)$.

Calendar entries

Only objects associated with calendar entries via the `iCal:summary` and `iCal:location` relationships are considered for profiling purposes. Concepts annotated by GATE as *Organization*, *Person*, or *Address* are matched to Wikipedia entries in the same manner as above and subsequently added to $\rho(u)$.

GPS information

An attempt is made to match any GPS information found in a Semantic Log to toponyms. The social networking site Plazes, [68], is used to accommodate this process. In Plazes, users are allowed to post details of locations they have been, or plan to be in. A location is annotated with GPS coordinates, either manually by the user or by virtue of being placed on a map. Each set of coordinates mapped to toponyms by Plazes are used to query Wikipedia. Entries retrieved at 100% relevance are then added to the user's profile.

Web browsing activity

URLs that the user has previously visited are treated in the same way as addresses found in e–Mail correspondence or iCal entries, described above.

#### 5.4.1.2 Online information

As described earlier, information held in social networking sites the users are subscribed with is imported into their Semantic Log. This section provides details on how such information is mapped to Wikipedia entries in order to be used for profiling purposes.

Last.fm

Artists a user has listened to are added to $\rho(u)$, and matched to Wikipedia articles about them via the Musicbrainz KB, much in the same way as in Section 4.3.1.

Flickr

Two aspects of the information held by Flickr have been identified as potentially useful for profiling purposes. First, the tags provided by a user for the pictures they upload are matched to Wikipedia entries using the process described in Section 4.1. Second, any 'geo–tagged' (associated with GPS coordinates) pictures are identified and an attempt is made to match such coordinates to toponyms through the Plazes API. The matched tags and retrieved toponyms are subsequently added to $\rho(u)$, the user's profile.

del.icio.us

In the case of del.icio.us, only tags are processed to obtain profiling features, as only a small number of bookmarked URLs can be expected to correspond directly to Wikipedia entries. While the URLs themselves are considered too specific for the task, the tags associated with them are thought of as high level descriptors of the information held at the bookmarked location. In the light of these empirical observations, any tags provided by a user in del.icio.us are mapped to Wikipedia entries (via applying the process outlined in Section 4.1) and added to $\rho(u)$. In addition, any tags provided by other del.icio.us subscribers for URLs the user has bookmarked are also included in this process.

#### 5.4.1.3 Producing recommendations using the profiles generated by the Semantic Logger

Although the SL system is fully implemented, it is an academic prototype, and efforts made to encourage new users to subscribe to the system have not been successful. The system holds over 100 accounts; however, only a single user had provided information from a variety of sources for an extended time period. As such any user studies or off–line experiments to assess the utility of the recommendations made in meeting user needs could not be carried out.

Instead, we set out to evaluate the suitability of each information source for the task of identifying user interests across multiple domains. To do so, the profiling features obtained for each source are used as input to produce recommendations for resources from each of the three datasets presented in the previous chapter: Netflix, Last.fm and del.icio.us.

Figure 5.11 shows the maximum posterior probability, $P(r_i^w|\varphi^*(\rho(u)))$, obtained for resources in each domain, when profiling resources from each source are used. In addition, the figure provides a threshold for each domain above which resources can be expected to be true positives. To calculate these thresholds, 100 experts have been randomly sampled from each domain. One resource was then removed from each expert's profile and the rest used as input to the recommendation algorithm. The average (over the 100 experts) posterior probability associated with the removed resource is used as the threshold. The process was repeated a number of times with a different set of experts, to ensure that the thresholds are consistent.

The user's profile was split in seven subsets, based on the source of information processed to identify each resource in the profile:

1. e–Mail: Resources obtained by applying named entity recognition to e–Mail correspondence
2. iCal: Similarly obtained resources from calendar entries.
3. GPS: Toponyms associated with GPS trackpoints logged.
4. Local: The union of the previous three sets of resources.
5. Last.fm (Top 50): The user's 50 most frequently played artists.
6. Last.fm (All): The set of all artists the user has listened to.
7. All: The entire user profile.

Each group of bars gives the maximum $P(r_i^w|\varphi^*(\rho(u)))$ for each domain in the order: Netflix (blue), Last.fm (green), and del.icio.us (red).

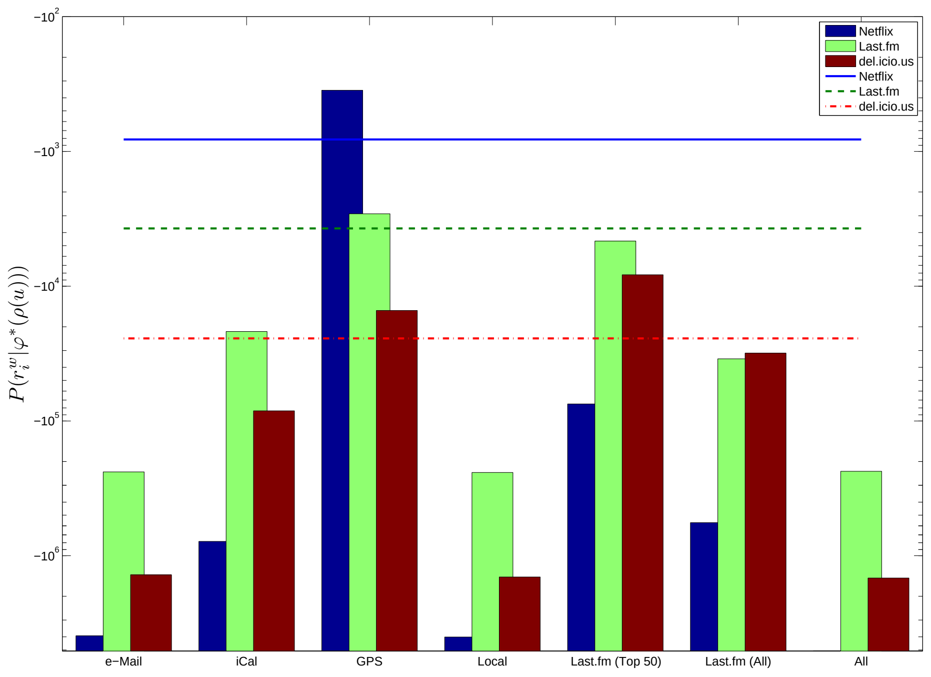


FIGURE 5.11: Maximum $P(r_i^w|\varphi^*(\rho(u)))$ for domains Netflix, Last.fm and del.icio.us, when $\rho(u)$ is restricted to each information source. The 'Local' subset contains any resources obtained by processing e–Mail correspondence, calendar entries, and GPS trackpoints. The maximum posterior obtained when using the user's profile in its entirety is given by the 'All' group. The solid blue line provides the threshold on $P(r_i^w|\varphi^*(\rho(u)))$ above which resources from the Netflix domain can be expected to be true positives. Corresponding thresholds for Last.fm and del.icio.us are given by the dashed green, and the dot–dashed red lines.

The first thing to observe is that the considerably larger 'e–Mail', 'Local', and 'All' sets score significantly lower than all others. This can be explained by revisiting our definition of Likelihood in Section 5.2.2.

Recall that:

$$P(\varphi^*(\rho(u))|r_i^w) = \sum_j P(f_j|r_i^w) \prod_k P(f_k) \tag{5.38}$$

such that $f_j \in \varphi^*(\rho(u)) \cap \varphi(r_i^w)$ , $f_k \in \varphi^*(\rho(u)) \setminus \varphi^*(\rho(u)) \cap \varphi(r_i^w)$. That is the probability of reaching **any one** of the elements of $\varphi^*(\rho(u))$ that are connected to $r_i^w$ and **all** the remaining elements, starting from the node $r_i^w$.

The reader is reminded of the discussion associated with Table 5.2, in Section 5.2.2. There, the necessity of penalising resources for the profiling features they are not connected was justified by showing that without doing so the system becomes biased towards producing recommendations that are uninformative and generic.

Now, as profiles grow larger, so does the set of unconnected profiling features per resource, $\varphi^*(\rho(u)) \setminus \varphi^*(\rho(u)) \cap \varphi(r_i^w)$. In contrast, the set of connected features for each resource can at most contain as many elements as the number of edges it participates in. Thus, for very large profiles the product of the priors of all unconnected features will vary very slightly across different resources, as the sets of unconnected features would consist mostly of the same elements. Further, if all resources are penalised equally, the likelihood is effectively calculated based on the connected features alone. For this reason, similar effects to those discussed in Section 5.2.2 can be expected to occur when extremely large profiles are used.

Moreover, the posterior probabilities obtained for domain resources based on profiling features from the 'GPS' set appear to mirror the thresholds obtained for each domain. This is not surprising as 'GPS' is the smallest of the subsets constructed, consisting only of a few points in Southampton and London. As these locations can be expected to share a considerable number of links, the identification of resources that connect to a large portion of such nodes becomes easier.

It is also interesting to compare results from the 'Last.fm (Top 50)' and 'Last.fm (All)' sets. For the reasons discussed above, the larger set was expected to score lower in terms of posterior probabilities. In addition, note that the threshold in the Last.fm domain had been obtained by assuming that user profiles consist only of such 'Top 50' lists for convenience in computational requirements. Had the entire profiles been used we would expect the threshold to fall lower. However, since the number of artists in each profile would vary much more, such a threshold would be less consistent.

Finally, it is interesting to note that although the threshold obtained for producing reliable recommendations in the Netflix domain is the highest, the posterior probabilities associated with recommendations are consistently the lowest, with the exception of those obtained based on the 'GPS' set. In contrast, Last.fm resources are recommended with higher probabilities than ones in del.icio.us for all sets except 'Last.fm (All)', consistent with the ordering of the thresholds obtained for the two domains. This is interpreted to indicate that the profiling features collected are less useful in recommending Netflix resources than for the other two domains.

#### 5.4.1.4 Context manipulation

The context of a recommendation is considered to be the user's situation at the time the need for a recommendation arose. This section discusses several mechanisms through which users can specify the characteristics of such situations at run–time, altering the recommendations they receive from the system.

As the captured information is integrated using ontologies and stored in a queryable knowledge base, users are provided with a flexible tool to define subsets of their profile they judge relevant to their current recommendation needs.

**Selecting a recommendation domain**

A simple, yet effective way to ensure that the recommendations provided reflect the users' current needs is by asking them to select the domain from which recommendations must originate. For example, consider someone who would like to discover new music, matching their taste. While it may be the case that the top candidates for recommendation from another domain have consistently higher posteriors given the user's profile than resources in the music domain, such recommendations should not be made as they would be of little use to the user.

**Selecting a subset of the profile**

Placing restrictions on which domains resources must belong to in order to be recommended will certainly have an impact on the final recommendations made. However, one cannot assume that this simplistic mechanism is sufficient to capture a wide variety of contexts relating to the user's recommendation need. Users may judge that certain subsets of their profile are irrelevant to the recommendation need at hand. As such, this section discusses various methods through which users may specify the profiling elements they wish to be considered by the system in producing recommendations.

Specifying the types of profiling features to consider

In some cases, only certain types of information held in user profiles can be considered relevant to a recommendation need. For example, a user who happens to be a music enthusiast may be seeking recommendations for locations on the Web that contain information related to his area of work. While receiving recommendations for Web locations alone can be achieved by specifying the recommendation domain, the large number of music–related elements in their profile may dominate over work–related elements, causing the system to recommend websites related to music. In order to rectify this situation, one may choose to exclude any profiling elements to do with music, or even specify that only features obtained from analysing e–Mail correspondence may be used.

Specifying a time interval

As we know from personal experience, people's interests change over time. It is also not uncommon for people to re–visit interests they held in the past, for a number of reasons. In order to accommodate for such situations, users are able to specify a time interval, such that only profiling elements created during the interval are considered. This approach is beneficial not only in capturing past interests, but also when users find that their sphere of interest is shifting to a new area. In the latter situation, they may specify that only the latest additions to their profile are considered to make recommendations.

Specifying a location

Users may sometimes require recommendations based on activities they undertook while being at a certain location. As it stands, locations may be specified in two ways. First, a set of GPS coordinates can be provided, along with the maximum distance of all points that can be considered as the same location. In this case, any GPS information corresponding to the specified area is first identified and used to compile a list of time intervals. This is carried out by considering points in the selected area that have been recorded in less than a day from each other as belonging to the same interval. Any profiling elements created during these intervals appear in the restricted profile. Plazes is then used to identify toponyms associated with the selected points, if possible. e–Mail messages and calendar entries found to refer to such toponyms are also included. Further, if calendar entries can be found, the restricted profile will then include any profiling elements that occur within the time intervals the entries span over. Instead of specifying GPS coordinates, users may choose to specify the name of a location. Essentially the same process is carried out, only here Plazes is used first, in order to map the provided toponyms to GPS coordinates.

Specifying a person

People can either be named, or identified by their e–mail address. In the former case, calendar entries associated with the name through the `ical:attendee` and `ical:organizer` relationships are identified. The time intervals such entries span over can then be used to obtain a restricted profile, as above. If an e–mail address is provided instead, the restricted profile will consist only of elements obtained through processing e–mail exchanges concerning that address. The reason for excluding all other sources of information, instead of using the messages obtained to construct corresponding time intervals, is that the e–mails people receive do not necessarily correspond to their activities at the time.

Specifying an event

Users are able to specify particular events, by providing the object of the `ical:summary` relationship, for calendar entries. Once matching entries have been identified, the profile is restricted based on the time intervals associated with the entries.

Specifying a type of activity

The `ical:categories` relationship provides another effective way of using calendar entries to segment user profiles. The range of this relationship is a list of keywords, such as `Holiday`, `Business`, `Anniversary`, etc. . By allowing users to select which category better reflects their situation at the time, their profiles can be segmented based on the time intervals associated with calendar entries that belong to this category.

The methods discussed in this section are in no way mutually exclusive for the task of defining the context of a recommendation. On the contrary, they may be combined as users see fit in order to manipulate their profile to reflect their current recommendation needs.

However, since sufficient information for evaluation has only been available for one person rules a user based study to assess if profiles restricted by contextual attributes can lead to more targeted recommendation.

Most of the methods of specifying a recommendation context presented here share in common the identification of an event, and a corresponding time interval. The Last.fm dataset was seen as a good opportunity to carry out some assessment of the effects of context manipulation, as the music listening behaviour of Last.fm subscribers is archived, and a 'chart' summarising of each week made available.

Thus, we can assess whether changes in music listening behaviour across two randomly selected weeks, for the same people, incur corresponding changes in the recommendation lists provided.

To carry this out, the Last.fm dataset was sampled to obtain a subset of 100 users. Then, for each one the list of available weeks was obtained from Last.fm, and two weeks chosen at random, to construct profiles $\rho'(e_i^1)$ and $\rho'(e_i^2)$, respectively. Corresponding recommendation lists for Last.fm resources, $rec_d(\rho'(e_i^1))$ and $rec_d(\rho'(e_i^2))$ are then obtained using the framework.

Spearman's Footrule is then used to compare the two profiles for each expert:

$$F^l(\rho'(e_i^1), \rho'(e_i^2)) = \sum_{i \in \rho'(e_i^1) \cup \rho'(e_i^2)} |rank(r_i, \rho'(e_i^1)) - rank(r_i, \rho'(e_i^2))|, \tag{5.39}$$

The constant $l$ is set to one larger than the size of the list the function *rank* operates on. $F^l(\rho'(e_i^1), \rho'(e_i^2))$ is then normalised so that if the two lists contain no items in common the Footrule is 1.

One of the profiles is then compared to the list of recommendations produced based on the other. Write:

$$F(\rho'(e_i^2), rec_d(\rho'(e_i^1)))) \tag{5.40}$$

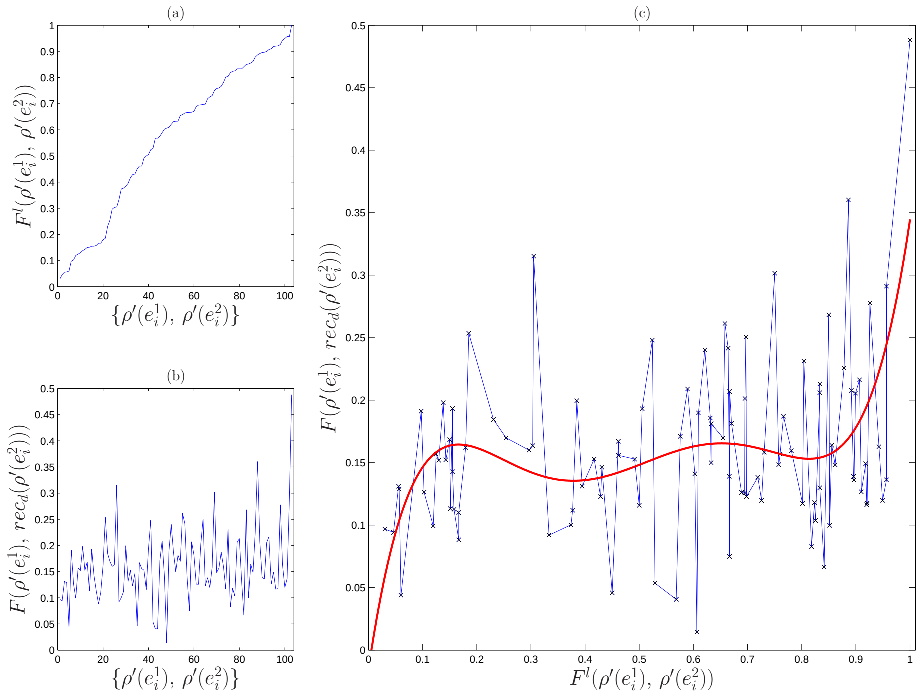


FIGURE 5.12: (a) Normalised Spearman's Footrule scores for the two profiles associated with each expert. Experts are sorted in ascending order of Footrule scores.
(b) Normalised Spearman's Footrule for the profiles $\rho'(e_i^1)$ and recommendation lists $rec_d(\rho'(e_i^2))$. Points are sorted in the same order as in (a).
(c) $F(\rho'(e_i^1), rec_d(\rho'(e_i^2)))$ as a function of $F^l(\rho'(e_i^1), \rho'(e_i^2))$. The red curve is a $5^{th}$ order polynomial fitted to the data.

and

$$F(\rho'(e_i^1), rec_d(\rho'(e_i^2)))) \tag{5.41}$$

Note that the constant $l$ becomes redundant, as every resource in the domain is ranked by $rec_d(\rho'(e_i^j))$. Thus the Footrule is calculated for all the resources in $\rho'(e_i^j)$.

The recommendations produced for $\rho'(e_i^2)$ are expected to be closer to $\rho'(e_i^1)$ when the two profiles are similar, giving a low Footrule score. Figure 5.12 provides the analysis of the relationship between $F^l(\rho'(e_i^1), \rho'(e_i^2))$ and $F(\rho'(e_i^1), rec_d(\rho'(e_i^2)))$ in a similar way as in Section 4.3.

Figures 5.12(a) and (b) give the footrule scores for the same ordering of experts $e_i$. Although 5.12(b) is very noisy, a positive gradient can be detected. Figure 5.12(c) presents the Footrule score for the first weekly profile for each $e_i$ and the recommendation list produced for the second, as a function of the corresponding score between the two profiles. As only a small number of points are available here, a sliding window approach could not be applied to reduce noise, as it could severely alter the data. Instead the figure also provides a $5^{th}$ order polynomial fitted to the data. It should be noted that

$$F(\rho'(e_i^2), rec_d(\rho'(e_i^1))) \approx F(\rho'(e_i^1), rec_d(\rho'(e_i^2))) \tag{5.42}$$

giving indistinguishable curves when plotted against $F^l(\rho'(e_i^1), \rho'(e_i^2))$. In addition, the coefficients for the fitted curve vary very slightly when different samples of experts are used. The order of the polynomial to fit was chosen by maximising the ratio between the *goodness–of–fit* and the number of free variables.

The graph shows evidence of correlation between the two scores when they are either very low, or very high. In other words, when the two weekly profiles consist of the same resources, ranked in a similar way, the ranking of the resources in the second profile are is correct in the recommendation list provided based on the first one. At the other extreme, when the two profiles differ greatly, the ranking of resources in the second profile will not be retrieved by comparing the posteriors assigned to the same resources based on the first one.

Moreover, differences in the two profiles of the magnitude $0.2 \leq F^l(\rho'(e_i^1), \rho'(e_i^2)) \leq 0.8$, seem to have a small effect on $F(\rho'(e_i^1), rec_d(\rho'(e_i^2)))$ as the datapoints vary relatively less, and remain low. One could speculate that in most such cases, the framework is relatively successful in retrieving the ranking in $\rho'(e_i^2)$ by making recommendations for $\rho'(e_i^1)$.

The recommendations produced by the framework for the music Last.fm users played during one week were then compared to the music they actually listened to during another. They were found to reflect the second week much better when the music played during both weeks was similar, than when the overlap between the two weeks was very small. This is seen as evidence to the claim that changes in user profiles incur corresponding changes in the recommendations produced. Moreover, the recommendation lists appear to reflect the music played in the second week well, for a wide range of moderately similar weekly 'charts', demonstrating the success of the framework to correctly capture user interests.

## 5.5 Conclusions

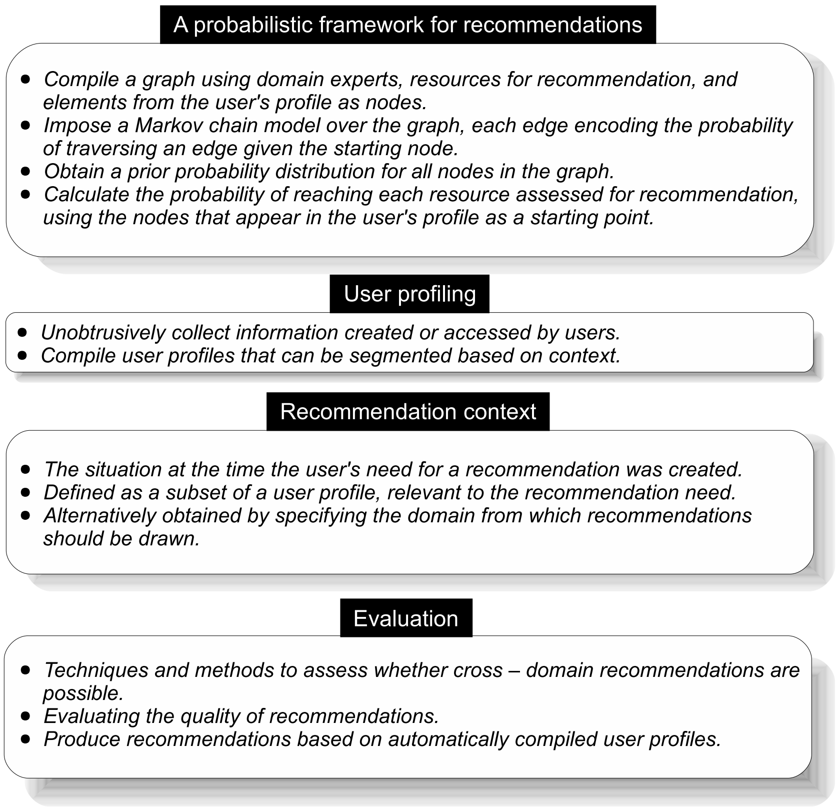

A probabilistic framework for recommendations
• Compile a graph using domain experts, resources for recommendation, and elements from the user's profile as nodes.
• Impose a Markov chain model over the graph, each edge encoding the probability of traversing an edge given the starting node.
• Obtain a prior probability distribution for all nodes in the graph.
• Calculate the probability of reaching each resource assessed for recommendation, using the nodes that appear in the user's profile as a starting point.
User profiling
• Unobtrusively collect information created or accessed by users.
• Compile user profiles that can be segmented based on context.
Recommendation context
• The situation at the time the user's need for a recommendation was created.
• Defined as a subset of a user profile, relevant to the recommendation need.
• Alternatively obtained by specifying the domain from which recommendations should be drawn.
Evaluation
• Techniques and methods to assess whether cross – domain recommendations are possible.
• Evaluating the quality of recommendations.
• Produce recommendations based on automatically compiled user profiles.

FIGURE 5.13: A visual representation of the contributions in this Chapter.

This chapter has provided details on the machine learning component of the recommendation framework presented in this dissertation. As the datasets RSs typically deal with are very large, the application of computationally intensive algorithms on the raw data becomes unfeasible. Thus, a sequence of pre–processing steps has been detailed to reduce the size of the dataset, enabling the application of such algorithms.

The dataset is first sampled using a number of statistical criteria in order to ensure that the sampling process does not alter the intrinsic structure of the ratings matrix. These are presented in Section 5.1.1. Ratings are then normalised for each expert that appears in the sample, giving a more objective view of the expert's relative preferences over the resources they have rated. Subsequently, the dataset is summarised using a *mrkd*–tree (presented in Section 5.1.3) by effectively merging similar expert profiles together.

The previous chapter showed how domain specific subgraphs can be extracted from Wikipedia, and used to obtain new representations for domain resources, which can in turn be used to express the same type of similarity between resources as that obtained through the comparison of rating vectors. Section 5.2 then presented a method to integrate such graphs with the preferences of domain experts on corresponding resources. Moreover, a Bayesian framework is imposed over the graph via a Markov chain [50] defined by a conditional probability associated with the traversal of each edge in the graph. The entries of the leading eigenvector of the probabilistic adjacency matrix associated with the graph are assumed to give an estimate of the prior probability distribution over all nodes in the graph, as supported by the spectral decomposition theorem.

Furthermore, Section 5.2.2 provided the definition for the likelihood – the conditional probability for reaching nodes in the graph that represent the user's profile, given a resource assessed for recommendation. The definition was then justified through the use of an illustrative example.

The probabilities defined are then used to calculate the probability of reaching each resource assessed for recommendation in the graph, given then nodes that appear in the input. The three available datasets, Netflix, Last.fm, and del.icio.us were then used in Section 5.3, to provide a qualitative assessment of the algorithms' performance. The algorithm was first used to provide recommendations in each of the three domains, using profiles from the same domain. This was carried out in order to obtain a baseline performance, to which the quality of cross–domain recommendations could be compared and contrasted. While the framework was found able to produce some meaningful recommendations in each domain, Netflix recommendations were in general more accurate than Last.fm ones, while recommendations in the del.icio.us domain were the least accurate of the three.

Most of the methods of specifying a recommendation context presented in Section 5.4.1.4 share in common the identification of an event, and a corresponding time interval. The Last.fm dataset was used to assess whether restricting the set of profiling features based on such attributes has a corresponding effect on the which resources are recommended.

Recommendations were produced by the framework for Last.fm users, using the music they played during a single week. They were then compared to the music they actually listened to during a second. The recommendations were found to reflect the second week much better when the music played during both weeks was similar, than when the overlap between the two weeks was very small. This fact supports the claim that changes in user profiles incur corresponding changes in the recommendations produced. Moreover, the framework appears successful in correctly capturng user interests, as recommendation lists appear to reflect the music played in the second week well, for a wide range of moderately similar weekly 'charts'.

The fact that Last.fm and del.icio.us share 1 965 subscribers provided a unique opportunity to evaluate the system's performance in producing cross–domain recommendations. Thus, the del.icio.us profiles for such users were used in Section 5.3.2 to produce Last.fm recommendations, and vice versa. Moreover, as labeled data from other domains had not been available for Netflix subscribers, this domain is used as a mediator between the other two in a further two experiments. In the first one, Last.fm profiles are first used to produce Netflix recommendations, which are subsequently used to produce del.icio.us recommendations. The second experiment expresses the same intuition in the reverse order: Netflix recommendations based on del.icio.us profiles are used as input to obtain recommendations for Last.fm resources.

The performance of the system in these four experiments was found to be very similar to its corresponding performance in producing single domain recommendations for the final recommendation domain in each experiment. Moreover, the predictive accuracy of the algorithm varies very slightly in the latter two experiments (where Netflix is used as an intermediate domain) than when recommendations between Last.fm and del.icio.us are produced directly. This indicates that the Netflix recommendations in both cases capture the information present in the original profiles very well. In addition, recommendations for del.icio.us resources were found to be the most accurate when Netflix is used as an intermediate domain, even in comparison to the algorithm's single–domain recommendation accuracy in the same domain.

Thus, it is concluded that cross–domain recommendations using the framework presented herein, are not only feasible but can be expected to be of similar quality to those produced by only considering a single recommendation domain. Further, as shown by our evaluation of recommendations from the del.icio.us domain, the adoption of a cross–domain recommendation paradigm by a RS can even prove beneficial in terms of predictive accuracy over recommendations using profiling features from the same domain.

# Chapter 6

# Conclusions

The central thesis of this dissertation is that:

**A framework can be developed to enable Recommender Systems to consider multiple domains from which recommendations may be drawn. Moreover, as a consequence of their cross domain nature, such systems will provide recommendations that are of high quality, unexpected, and geared solely towards satisfying user needs.**

as presented in Chapter 1. In order to develop such a framework, two main requirements have been identified:

1. The availability of a universal vocabulary of terms and relationships between them, such that any possible resource can be described using terms from this vocabulary. This is deemed essential by the need to assess similarity between resources that appear in different domains. The use of Wikipedia for this purpose is detailed in Chapter 4.
2. A machine learning component apt for the task of producing cross–domain recommendations. This is provided in Chapter 5, where a probabilistic model for the task, and a corresponding graph–based probabilistic method to assess resources for recommendation are defined.

Here, in the last chapter of this dissertation, a summary of the experimental results obtained in chapters 4, 5, and 5.4 is provided, detailing their contributions to the central thesis. In addition, an overview of the research questions that remain open, and an outline of directions for future work are provided in the last section of this chapter.

## 6.1 Universal Vocabulary

Wikipedia has been successfully used in the role of the universal vocabulary in our approach. This was justified in Chapter 4, by showing that the similarity between domain resources (assessed using rating vectors) is preserved when projected vectors corresponding to resources, obtained from Wikipedia, are compared instead. Moreover, this success is demonstrated by the high quality of the recommendations produced in Chapter 5, using the projected representations.

Three different modes of obtaining Wikipedia articles representative of domain resources were identified. Each mode applies to a different set of domain characteristics, providing a comprehensive toolset that is expected to be applicable to the majority of available domains. These are:

- Through the availability of explicit, high quality mappings between domain resources, and Wikipedia articles. For example, the Musicbrainz KB provides such mappings for resources in the Last.fm domain.
- By processing the categories the articles whose labels match those of domain resources belong to, to assess whether they refer to the same concepts as the resources. In the domain of film recommendations, for example, articles which do represented domain resources can be expected to fall under a small number of categories, such as: *'Film'*, *'Series'*, *'TV'*, *'DVD'*, *'Episode'*, or *'Show'*.
- By processing tags associated with resources to identify corresponding Wikipedia articles. In the case of del.icio.us, Wikipedia could not be expected to contain an article corresponding to each bookmarked URL. Instead, using the work published by Szomszor et. al. [84], tags which could be mapped to Wikipedia without requiring any disambiguation were identified. The corresponding articles were then used to obtain Wikipedia projections of domain resource, based on the tags assigned to them.

In addition, the evaluative methodology applied to assess the utility of Wikipedia as the universal vocabulary can naturally be applied to any corpus candidate for that role. Moreover, if other such corpora become available, the same method of assessment can be used to identify which one is likely to perform best.

## 6.2 Probabilistic model and algorithm

Chapter 5 then presented the specifics of the probabilistic model used to produce recommendations from multiple domains. First, it provided a technique to integrate expert preferences on domain resources and the relationships between projections of the same resources in the universal vocabulary in a single graph.

A Markov chain was then imposed over the graph, to give natural definitions for the probabilities associated with the traversal of each edge in the graph, as well as for the prior probability distribution over all nodes. A resource is thus assessed for recommendation by calculating the probability of reaching its corresponding node in the graph, given the nodes in the user profile used as input to the algorithm as a starting point.

The three available datasets, Netflix, Last.fm, and del.icio.us were then used to provide a qualitative assessment of the algorithms' performance. Recommendations in each of the three domains, using profiles from the same domain, were produced first in order to obtain a baseline performance, to which the quality of cross–domain recommendations could be compared and contrasted. While the framework was found able to produce some meaningful recommendations in each domain, Netflix recommendations were in general more accurate than Last.fm ones, while recommendations in the del.icio.us domain were the least accurate of the three.

The fact that Last.fm and del.icio.us share 1 965 subscribers provided a unique opportunity to evaluate the system's performance in producing cross–domain recommendations. Thus, the del.icio.us profiles for such users were used in Section 5.3.2 to produce Last.fm recommendations, and vice versa. Moreover, as labeled data from other domains had not been available for Netflix subscribers, this domain is used as a mediator between the other two in a further two experiments. In the first one, Last.fm profiles are first used to produce Netflix recommendations, which are subsequently used to produce del.icio.us recommendations. The second experiment expresses the same intuition, in the reverse order: Netflix recommendations based on del.icio.us profiles are used as input to obtain recommendations for Last.fm resources.

The performance of the system in these four experiments was found to be very similar to it's corresponding performance in producing single domain recommendations for the final recommendation domain in each experiment. Moreover, the predictive accuracy of the algorithm varies very slightly in the latter two experiments (where Netflix is used as an intermediate domain) than when recommendations between Last.fm and del.icio.us are produced directly. This indicates that the Netflix recommendations in both cases capture the information present in the original profiles very well. In addition, recommendation for del.icio.us resources were found to be the most accurate when Netflix is used as an intermediate domain, even in comparison to the algorithm's single–domain recommendation accuracy in the same domain.

Thus, it was argued that cross–domain recommendations using the framework presented herein, are not only feasible but can be expected to be of similar quality to those produced by only considering a single recommendation domain. Further, as shown by our evaluation of recommendations from the del.icio.us domain, the adoption of a cross–domain recommendation paradigm by a RS can even prove beneficial in terms of predictive accuracy over recommendations using profiling features from the same domain.

In summary, this dissertation has provided evidence towards the claims that each of the main requirements identified in the beginning of this chapter have been fulfilled by the proposed framework.

## 6.3 Directions for future work

As is the nature of research, a number of questions were raised when various aspects of the work presented here was being carried out. Due to time constraints not all have been addressed. This section provides an overview of issues that warrant further examination.

While Wikipedia was judged adequate for use as the universal vocabulary in our approach, it is far from an ideal match to the definition. A number of problems have been identified in Sections 5.2.2 and 5.4, relating to the presence of articles which carry little information, but are very well connected to articles that represent domain resources. This problem was found to be most acute when user profiles are very large, as such uninformative articles can be the only ones the majority of profiling elements shares connections to.

One way to address the problem, would be to identify such Wikipedia articles, and exclude them from the recommendation process. While a number of metrics can be used to support the identification of uninformative articles, the selection of method and corresponding thresholds must be carried out carefully, as potentially valuable information could easily get discarded.

In an attempt to rectify the issue, the initial definition of likelihood (the conditional probability for reaching nodes in the graph that represent the user's profile, given a resource assessed for recommendation) was revised. However, the system was found to suffer from the same effects using the revised definition when extremely large profiles were used. Therefore, it could be beneficial to revise the definition further by adding a parameter which would depend on the size of the profile used.

A great amount of emphasis is placed here on the size of the profiles used. It was speculated that by allowing users to select small subsets of their profiles, based on contextual attributes specific to their current recommendation needs, would alleviate the effects of this issue. This claim has not been evaluated directly however, as a sufficient number of Semantic Logger users had not been available. However, the recommendation component of the system had not been available when users were first requested to subscribe. Now that it is, it would be prudent to make large efforts in promoting the system to potential users, as it may provide an added incentive for people to subscribe.

If that is successful, the collected data can then be used to assess the validity of the speculations made.

# Bibliography

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *j-TOIS*, 23(1):103–145, January 2005.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[3] H. Alani, S. Dasmahapatra, K. O'Hara, and N. R. Shadbolt. Identifying communities of practice through ontology network analysis. *IEEE Intelligent Systems*, 18(2):18–25, 2003.

[4] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–25, 1951.

[5] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.

[6] J. Bennet. The case of netflix. In *Recommenders06.com*, pages 1–21, Bilbao, 2006.

[7] A. Berenzweig, B. Logan, D. Ellis, and B. Whitman. A large-scale evaluation of acoustic and subjective music similarity measures. In *4th International Symposium on Music Information Retrieval*, 2003.

[8] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. Cross-domain mediation in collaborative filtering. In *11th International Conference on User Modeling*, pages 355–359, Corfu, Greece, June 2007.

[9] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5), May 2001.

[10] Krishna Bharat, Tomonari Kamba, and Michael Albers. Personalized, interactive news on the web. *Multimedia Syst.*, 6(5):349–358, 1998.

[11] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *15th International Conference on Machine Learning*, pages 46–54. Morgan Kaufmann, San Francisco, CA, 1998.

[12] Derek G. Bridge and Alex Ferguson. Diverse product recommendations using an expressive language for case retrieval. In *ECCBR '02: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning*, pages 43–57, London, UK, 2002. Springer-Verlag.

[13] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[14] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[15] Iván Cantador, Martin Szomszor, Harith Alani, Miriam Fernández, and Pablo Castells. Enriching ontological user profiles with tagging history for multi-domain recommendations. In *1st International Workshop on Collective Semantics: Collective Intelligence & the Semantic Web (CISWeb 2008)*, June 2008.

[16] Chris Chatfield. Apples, oranges and mean square error. *International Journal of Forecasting*, 4(4):515–518, 1988.

[17] H. Cunningham. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254, 2002.

[18] F. Dawson and D. Stenerson. Internet calendaring and scheduling core object specication. request for comments 2445. Technical report, Internet Engineering Task Force, 1998.

[19] S.C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[21] K. Deng and A. Moore. Multiresolution instance-based learning. In *Proceedings of the Twelfth International Joint Conference on Artificial Intellingence*, pages 1233–1239, San Francisco, 1995. Morgan Kaufmann.

[22] Pedro Domingos and Michael J. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *International Conference on Machine Learning*, pages 105–112, 1996.

[23] David Dupplaw, Madalina Croitoru, Antonis Loizou, Srinandan Dasmahapatra, Paul Lewis, Mischa Tuffield, and Liang Xiao. Multimedia markup tools for openknowledge. In *1st Workshop on Multimedia Annotation and Retrieval enabled by Shared Ontologies, International Conference on Semantics and Digital Media Technologies*, December 2007.

[24] David S. Emmerich. Signal detection theory and psychophysics. *The Quarterly Review of Biology*, 42(4), 1967.

[25] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. *SIAM J. Discret. Math.*, 17(1):134–160, 2003.

[26] Carl Friedrich Gauss. *Theoria motus corporum coelestium: in sectionibus conicis solem ambientium / Theory of the motion of the heavenly bodies moving about the sun in conic sections.* Sumtibus F. Perthes et I.H. Besser, Hamburgi, 1809.

[27] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: an automatic citation indexing system. In *ACM Conference on Digital Libraries*, pages 89–98. ACM Press, 1998.

[28] J. Giles. Internet encyclopaedias go head to head. *Nature*, 438(7070):900–901, December 2005.

[29] William Goffman and Vaun A. Newill. A methodology for test and evaluation of information retrieval systems. *Information Storage and Retrieval*, 3(1):19–25, 1966.

[30] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.

[31] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.

[32] G. Gonzalez, B. Lopez, and J. LL. de la Rosa. A multi–agent smart user model for cross-domain recommender systems. In *Beyond Personalization 2005: The Next Stage of Recommender Systems Research. International Conference on Intelligent User Interfaces IUI05*, San Diego, California, USA, January 2005.

[33] Google. Blogger. *https://www.blogger.com/*, 1999.

[34] Google. Google app engine: Run your web applications on google's infrastructure. *http://code.google.com/appengine/*, 2008.

[35] Google. Google docs: Create and share your work online. *http://docs.google.com*, 2008.

[36] P. R. Halmos. What does the spectral theorem say? *The American Mathematical Monthly*, 70(3):241–247, 1963.

[37] Stephen P. Harter and Thomas E. Nisonger. Isi's impact factor as misnomer: a proposed new measure to assess journal impact. *Journal of the American Society for Information Science*, 48(12):1146–1148, 1997.

[38] T. Hofmann. Probabilistic latent semantic indexing. In *22nd Annual ACM Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, California, 1999.

[39] Roman Hogg, Miriam Meckel, Katarina Stanoevska-Slabeva, and Robert Martignoni. Overview of business models for web 2.0 communities. pages 23–37, 2006.

[40] Hugh Glaser. Semantic squirrel special interest group. *http://www.semantic-squirrel.org/*, 2001.

[41] IMDb.com, Inc. The internet movie database. *http://www.imdb.com/*, 1990.

[42] C. Kadie J. Breese, D. Heckerman. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[43] Maurice Kendall and Jean D. Gibbons. *Rank Correlation Methods.* A Charles Griffin Title, 5 edition, September 1990.

[44] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[45] Hyung Joon Kook. Profiling multiple domains of user interests and using them for personalized web support. In *Advances in Intelligent Computing, International Conference on Intelligent Computing*, pages 512–520, Hefei, China, August 2005.

[46] Bruce Krulwich. Lifestyle finder: Intelligent user profiling using large-scale demographic data. *AI Magazine*, 18(2):37–45, 1997.

[47] O. Lassila and R. Swick. Resource description framework (RDF) model and syntax specification. *World Wide Web Consortium*, 1999.

[48] Last.fm Ltd. Last.fm. *http://www.last.fm*, 2006.

[49] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[50] Andrey Markov. *An Example of Statistical Investigation of the Text Eugene Onegin Concerning the Connection of Samples in Chains (1906).* David Link. Science in Context 19.4, 2006.

[51] Martin Porter. Snowball: A language for stemming algorithms. *http://snowball.tartarus.org/*, 2001.

[52] Lorraine McGinty and Barry Smyth. On the role of diversity in conversational recommender systems. In *Proceedings of the Fifth International Conference on Case-Based Reasoning*, pages 276–290. Springer, 2003.

[53] P. McJones. Eachmovie collaborative filtering data set. 1997.

[54] G. McLachlan and K. E. Basford. Mixture models. *Marcel Dekker, INC, New York Basel*, 1988.

[55] Sean Michael McNee. *Meeting User Information Needs in Recommender Systems.* PhD thesis, University of Minnesota –Twin Cities, June 2006.

[56] S. E. Middleton, D. C. De Roure, and N. R. Shadbolt. Capturing knowledge of user preferences: ontologies in recommender systems. In *1st international conference on Knowledge Capture*, pages 100–107, New York, NY, USA, 2001. ACM Press.

[57] B. Miller. *Toward a personallized recommender system.* PhD thesis, University of Minnesota –Twin Cities, June 2003.

[58] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[59] Nader Mirzadeh, Francesco Ricci, and Mukesh Bansal. Supporting user query relaxation in a recommender system. In Kurt Bauknecht, Martin Bichler, and Birgit Pröll, editors, *5th International Conference on E-Commerce and Web Technologies*, pages 31–40, Zaragoza, Spain, 2004. Springer.

[60] A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In M. Kearns and D. Cohn, editors, *Advances in Neural Information Processing Systems*, pages 543–549, 340 Pine Street, 6th Fl., San Francisco, CA 94104, April 1999. Morgan Kaufman.

[61] John Paul Mueller. *Mining Google Web Services: Building Applications with the Google API.* SYBEX Inc., Alameda, CA, USA, 2004.

[62] H. Jr. Murray. Methods for satisfying the needs of the scientist and the engineer for scientific and technical communication. *A Press Release*, 1966.

[63] Netflix, Inc. Netflix prize. *http://www.netflixprize.com/*, 2006.

[64] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[65] M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[66] Karl Pearson. Determination of the coefficient of correlation. *Science*, 30(757):23–25, 1909.

[67] D. M. Pennock, E. Horvitz, and C. L. Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *AAAI/IAAI*, pages 729–734, 2000.

[68] Plazes AG. Right plaze, right time. *http://plazes.com/*, 2008.

[69] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM Press.

[70] Francesco Ricci and Fabio del Missier. Supporting travel decision making through personalized recommendation. pages 231–251, 2004.

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

[72] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, pages 513–523, 1988.

[73] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167, 2000.

[74] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems - a case study. In *ACM WebKDD2000, Web Mining for E-Commerce - Challenges and Opportunities*, 2000.

[75] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[76] Badrul Munir Sarwar. *Sparsity, scalability, and distribution in recommender systems*. PhD thesis, University of Minnesota, 2001. Adviser-John T. Riedl.

[77] L. Saul and F. Pereira. Aggregate and mixed-order Markov models for statistical language processing. In Claire Cardie and Ralph Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 81–89. Association for Computational Linguistics, Somerset, New Jersey, 1997.

[78] J. Ben Schafer, Joseph A. Konstan, and John Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1-2):115–153, 2001.

[79] A. I. Schein, A Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *25th International ACM Conference on Research and Development in Information Retreival*, 2002.

[80] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Croc: A new evaluation criterion for recommender systems. *Electronic Commerce Research*, 5(1):51–74, 2005.

[81] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". In *CHI*, pages 210–217, 1995.

[82] Charles Edward Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15:72–101, 1904.

[83] A. Swartz. Musicbrainz: A semantic web service. *IEEE Intelligent Systems*, 17(1):76–77, 2002.

[84] Martin Szomszor, Harith Alani, Ivan Cantador, Kieron O'Hara, and Nigel Shadbolt. Semantic modelling of user interests based on cross-folksonomy analysis. In *7th International Semantic Web Conference (ISWC)*, October 2008.

[85] Martin Szomszor, Iván Cantador, and Harith Alani. Correlating user profiles from multiple folksonomies. In *ACM Confrence on Hypertext and Hypermedia*, June 2008.

[86] Technorati, Inc. Technorati. *http://technorati.com/*, 2006.

[87] The Wikimedia Foundation Inc. Wikipedia: The free encyclopedia. *http://wikipedia.org*, 2006.

[88] Marc Torrens, Boi Faltings, and Pearl Pu. Smartclients: Constraint satisfaction as a paradigm for scaleable intelligent information systems. *Constraints*, 7(1):49–69, 2002.

[89] M. M. Tuffield, A. Loizou, D. Dupplaw, S. Dasmahapatra, P. H. Lewis, D. E. Millard, and N. R. Shadbolt. The semantic logger: Supporting service building from personal context. In *Proceedings of the 3rd ACM Workshop on Capture and Archival of Personal Experience, ACM Multimedia*, 2006.

[90] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[91] J. J. Verbeek, J. R. J. Nunnink, and N. Vlassis. Accelerated EM-based clustering of large data sets. The Netherlands, 2005. Kluwer Academic Publishers.

[92] Michael J. Weiss. *The clustering of America*. Harper & Row, New York :, 1st ed. edition, 1988.

[93] Yahoo Inc. delicious. social bookmarking. *http://delicious.com/*, 2008.

[94] Yahoo Inc. flickr: Share your photos. watch the world. *http://www.flickr.com/*, 2008.

[95] W. Yang, Z. Wang, and M. You. An improved collaborative filtering method for recommendations' generation. In *IEEE International Conference on Systems, Man & Cybernetics*, pages 4135–4139, 2004.

[96] C. N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *World Wibe Web*, pages 22–32, 2005.