UNDERSTANDING PREFERENCES AND SIMILARITIES FROM USER-AUTHORED TEXT: APPLICATIONS TO SEARCH AND RECOMMENDATIONS

by

GAYATREE GANU

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Amélie Marian

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2014

ABSTRACT OF THE DISSERTATION

Understanding Preferences and Similarities from User-Authored Text: Applications to Search and Recommendations

by Gayatree Ganu

Dissertation Director: Amélie Marian

Users rely increasingly on online reviews, forums and blogs to exchange information, practical tips, and stories. Such social interaction has become central to their daily decision-making processes. However, user-authored content is in a free-text format, usually with very scant structured metadata information. Users often face the daunting task of reading a large quantity of text to discover potentially useful information. This thesis addresses the need to automatically leverage information from user-authored text to improve search and to provide personalized recommendations matching user preferences.

We first focus on developing accurate text-based recommendations. We the rich information present in user reviews by identifying the review parts pertaining to different product features and the sentiment expressed towards each feature. We derive text-based ratings which serve as alternate indicators of user assessment of the product. We then cluster similar users based on the topics and sentiments in their reviews. Our results show that using text yields better user preference predictions than those from the coarse star ratings. We also make fine-grained predictions of user sentiments towards the individual product features.

In the interactive and social forum sites users frequently make connections with other users, enabling them to find the right person to answer their questions. A challenge then, is to score and rank the short snippets of forum posts while taking into account these personal connections.

In this thesis, we learn user similarities via multiple indicators like shared information needs, profiles, or topics of interest. We develop a novel multidimensional model that uniformly incorporates the heterogeneous user relations in finding similar participants, to predict future social interactions and enhance keyword search.

Search over user-authored data like forums requires providing results that are as complete as possible and yet are focused on the relevant information. We address this problem by developing a new search paradigm that allows for search results to be retrieved at varying granularity levels. We implement a novel hierarchical representation and scoring technique for objects at multiple granularities. We also present a score optimization algorithm that efficiently chooses the best k-sized non-overlapping result set. We conduct extensive user studies and show that a mixed granularity set of results is more relevant to users than standard post-only approaches.

In summary, this thesis studies the problems in understanding user behavior from textual content in online reviews and forums. We present efficient techniques to learn user preferences and similarities to enhance search and recommendations.

Acknowledgements

We are a culmination of all our life's experiences. I want to thank every person and incident that culminated into my achievements. Then again, it is not really an end but just a new beginning.

First and foremost, I would like to thank my advisor Amélie Marian, without whose constant positive encouragement this research would not have been possible. Amélie has been a very understanding and kind mentor, and has always appreciated my ideas big or small, ambitious or naive. I have learned so many things from her: I have learned to hold my head up and be proud of every achievement, to understand priorities in work and in life, and to communicate and present my work. She also taught me to wander the unknown realms of research, learn new concepts and build tools to address the big problems.

Thanks to my thesis committee members Alex Borgida, Tina Eliassi-Rad and Daniel Tunkelang for taking the time to give me constructive feedback, discussing my research and their encouragement and support.

I would like to thank Branislav Kveton, who taught me that research can be fun. Everything does not always works out, but the important thing is perseverance and smiling through it all. Nothing ever felt impossible when working with Brano. I would like to thank Jean Bolot and everyone at Technicolor Research Lab for their support. I hope we have opportunities to work together again. Also, thanks to Osnat Mokryn for long brainstorming sessions and advice, and Yoann Bourse without whom I would not know the world of Nyan cat and My Little Pony.

Noémie Elhadad has been very influential in this work, from the inception of the projects, to data collection and grants.

I am extremely grateful to the Computer Science department at Rutgers University. Everyone on the staff: especially our graduate secretary Carol DiFrancesco made all administrative matters a breeze. Hanz Makmur and everyone at LCSR solved all computing problems with

expert ease. I would also like to thank Kate Goelz, the course co-ordinator when I taught Computer Applications for Business. I am sure I would have never pursued this career path if I had not received the financial support from her program. I am grateful for all the varied experiences, new friends and growing-up phases at the Graduate Students Association, International Students Office and the Werblin gym.

Over the years, several people have influenced my thinking and have helped me learn and grow. Thanks to Minji Wu, my lab-mate and close friend for 6+ years and also to the many members of the SPIDR lab and reading groups. I also thank Pallavi, Rekha, Nisha and other room-mates for many challenging discussions, long nights playing board-games and their invaluable friendship. I owe so much to my many friends and close confidants at Rutgers University: Mangesh, Shweta, Pravin and Arzoo, my undergraduate studies at SPCE and my childhood friend Deepti. They have seen me through all the ups and downs, and have always been a source of fun and support.

Family ties are my fuel and my deepest heart-felt thanks to my very loving family. My Dad's encouragement to dream big and my Mum's teaching to persevere and follow through on those dreams are the reasons for all my achievements. Thanks also to my brother Shailesh and Tanya, who will agree with me that we were uniquely blessed to have very supporting, clever and encouraging parents who gave us an amazing start. My husband Abhinav, to whom I owe every smile and happy memory in the last five years and without whom there would be no reason to strive to work hard and plan a future. He has been my greatest support throughout the many times when I was dejected and tired; this truly would not be possible without him. I also thank him for bringing me into his very large and loving family. The last few years would have been an impossible drag without the fun times with Abhinav's parents, Nitya and Pranav, and most of all my nieces Ananya and Anushka.

Dedication

To my husband Abhinav

and to my parents.

To learning for that reason alone.

To new beginnings, to new challenges . . .

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The recent Web 2.0 user-generated content revolution has enabled online users to broadcast their knowledge and experience. Web users have wholeheartedly incorporated peer-authored posts into their lives, whether to make purchasing decisions based on recommendations or to plan a night out using restaurant and movie reviews. Despite the growing popularity, there has been little research on the quality of the content. In addition, web sites providing user authored content are surprisingly technologically poor: users often have no choice but to browse through massive amounts of text to find a particular piece of interesting information.

A popular domain for online user generated content is reviews of products and services. Accessing and searching text reviews is frustrating when users only have a vague idea of the product or its features and they need a recommendation or closest match. Keyword searches typically do not provide good results, as the same keywords routinely appear in good and in bad reviews [10]. Yet another challenge in understanding reviews is that a reviewer's overall rating might be largely reflective of product features in which the search user is not interested. Consider the following example:

Example 1: *The NYC restaurant Lucky Cheng's in* Citysearch *(http://newyork.citysearch.com) has 65 user reviews of which 40 reviews have a 4 or 5 star rating (out of 5 possible stars). Majority positive reviews, however, praise the ambience of the restaurant, as shown in the following sentences extracted from the reviews:*

- *"obviously it's not the food or drinks that is the attraction, but the burlesque show"*
- *"Dont go for the food because the food is mediocore."*
- *"The food was okay, not great, not bad.[...]Our favorite part, though, was the show!"*

The negative reviews complain at length about the price and service. A user not interested

in ambience would probably not want to dine at this restaurant. However, a recommendation based on star ratings would label this restaurant as a high-quality restaurant.

User experience would be greatly improved if the structure of the content in reviews was taken into account, i.e., if review parts pertaining to different product features, as well as the sentiment of the reviewer towards each feature were identified. This information, coupled with the metadata associated with a product (e.g., location or cuisine for restaurants), can then be used to analyze and access reviews. However, identifying structured information from free-form text is a challenging task as users routinely enter informal text with poor spelling and grammar, as discussed in Chapter 2.

We propose techniques that harness the rich information present in the body of the reviews by identifying the review parts pertaining to different product features (e.g., food, ambience, price, service for a restaurant), as well as the sentiment of the reviewer towards each feature (e.g., positive, negative or neutral) and leverage this information to improve user experience. Our work addresses categorization and sentiment analysis at the sentence level as web reviews are short and designed to convey detailed information in a few sentences. We performed an in-depth classification of real-world restaurant review data sets using both supervised and semi-supervised techniques, and report on our findings in Chapter 3.

Ideally, users should not have to read through several reviews, but should be presented with items that they would find interesting or useful based on some notion of preference through similarity with other users or items. This task of preference matching is carried out by recommendation systems [21]. Current recommendation systems such as the ones used by Netflix or Amazon [74] rely predominantly on structured metadata information to make recommendations, often using only the star ratings, and ignore a very important information source available in reviews: the textual content. We apply our text analysis from Chapter 3 to a recommendation scenario in Chapter 4 and show that the rich textual information can improve rating prediction quality. In addition, we propose methods to predict the sentiments of users towards individual restaurant features and enhance user experience by presenting the review parts pertaining to these features.

Another popularly used platform to exchange information is online forums. A common

approach to gather feedback on a product, disease, or technical problem is to ask a question on an Internet forum and await answers from other participants. Alternatively, one can search through information in forums which often is already present as part of earlier discussions. Unfortunately, web forums typically offer only very primitive search interfaces that return all posts that match the query keyword. Because of the nature of keyword-based search, short posts containing the query keywords may be ranked high even if they do not have much useful information, while longer posts with relevant information could be pushed down in the result list because their normalized scores are penalized by their size. When issuing queries to forums, users face the daunting task of sifting through a massive number of posts to separate the wheat from the chaff.

As a new search problem, search over forum text yields interesting challenges. Background information is often omitted in posts as it is assumed that readers share the same background knowledge [34]. A critical challenge then for web forums search is to provide results that are as complete as possible and that do not miss some relevant information but that are also focused on the part of individual threads or posts containing relevant text. Therefore, in this type of search the correct level of result granularity is important. Dynamically selecting the best level of focus on the data helps users find relevant answers without having to read large amounts of irrelevant text. Therefore, our goal is to improve the experience of users searching through web forums by providing results that focus on the parts of the data that best fit their information needs. Our approach allows for search results to be returned at varying granularity levels: single pertinent sentences containing facts, larger passages of descriptive text, or entire discussions relevant to the search topics. We propose a novel multi-granularity search for web forum data to offer the most relevant information snippets to a user query, as described in Chapter 5.

Finding similarities in forum participants can enable a search system to retrieve more useful and relevant results authored by like-minded users. Finding such personalized similarity is a complex problem due to the mix of various signals of user affinity. Occasionally, web forums allow users to make explicit friendships, thus providing the social graph over users. Additionally, there exist several implicit cues of user affinity like participating in the same threads or discussing the same topics. Yet, two users having similar information needs at different times might never participate in the same discussions. For instance, in a forum for mothers several

participants will have similar questions about feeding, teething, and sleep patterns. However, some mothers with older children will never participate in newer threads related to infants, even though they have experience raising infants and may have participated in such threads in the past. On the other hand, for a location-based business search, forum participants in the query location are likely to provide answers despite largely varying profiles or topics of interest. Therefore, it is an important and challenging problem to uniformly capture similarities in online users while taking into account multiple implicit signals like user profiles, topics of interest or information needs.

Our approach to address the problem of finding like-minded forum participants is to use a multidimensional random walk that dynamically learns importance of the various inter-user relations. Random walk on graphs, where nodes represent the forum participants and edges represent the strength of node association or node similarity, correctly captures many notions of user similarity. However, existing random walk algorithms assume that the underlying graph is homogeneous comprising of nodes and edges of a single type each. Our work extends the random walk (RW) algorithm on a graph to a multidimensional scenario, where each dimension represents a different user relation semantic connecting the nodes. Our algorithm learns the importance of the various interpersonal relations for a user and finds the top-k most similar neighbors across these heterogeneous relations. Thus, we explore the implicit social interactions of forum users to enhance search and personalization of results as described in Chapter 6.

In summary, this thesis studies the problems in understanding online user behavior and similarities from the textual content in reviews and forums. We present efficient techniques to learn user preferences to enhance search and recommendations.

Chapter 2

Challenges in Automatic Processing of Online User Authored Text

In 2006, Time Magazine chose as its *Person of the Year* the millions of anonymous contributors of user-generated content. Products are routinely reviewed by customers on e-commerce destinations such as `amazon.com` and review-dedicated websites like `citysearch.com` and `tripadvisor.com`. Web users, for their part, rely strongly on peer-authored posts into their lives, whether to make purchasing decisions based on peer recommendations or to plan a night out using restaurant and movie reviews. According to surveys, online reviews are second only to word of mouth in purchasing influence [5]. Another study [4] shows that 86% of polled individuals find customer reviews extremely or very important. Furthermore, 64% of the individuals report researching products online often, no matter where they buy the product (Web, catalog, store, etc.).

If online reviews are a trusted and useful source of information for Web users, tools that leverage the valuable information present in reviews are lacking sorely. The sheer number of reviews available for a given product can be overwhelming for users trying to get a comprehensive view of reviewers' opinions. Furthermore, it is often impossible for users to know in advance which reviews contain information relevant to their specific information needs unless they skim all the reviews. Not surprisingly, 78% of polled individuals indicate they spend more than 10 minutes reading reviews for a given product type [4]. Popular websites have started to deploy techniques to aggregate the vast information available in user reviews and to identify reviews with high information content. `amazon.com`, for instance, relies on the star ratings to aggregate reviews and user votes to determine which reviews are helpful. The Internet Movie Database (`imdb.com`) uses the reviewer profile and demographics. Websites dedicated to particular products, like `citysearch.com` for restaurants, ask the reviewers a set of descriptive questions that can be used later as metadata information (Crowded, Trendy) when searching

products (Figure 2.1). This approach however, puts the burden of providing aggregate information on review contributors by asking them many binary or multiple choice questions. All these techniques ultimately depend on how much users are willing to contribute, either the reviewers themselves in rating products and answering descriptive questions or the review readers in rating the usefulness of a review. Furthermore, pre-determined metadata are not always flexible enough to represent all the information a user can contribute in a review. Unfortunately, most aggregation techniques so far, ignore the information conveyed in the text of a review.

Figure 2.1: Citysearch asks the reviewer several optional questions to try and gain information and opinions otherwise available only in text.

User experience would be greatly improved if the structure of the review texts were taken into account, i.e., if the review parts pertaining to different features of a product (*e.g.*, food, atmosphere, price, service for a restaurant) were identified, as well as the sentiment of the reviewer towards each feature (*e.g.*, positive, negative or neutral). This information, coupled with the metadata associated with a product (*e.g.*, location and type of cuisine for restaurants), could then be used to analyze, aggregate and search reviews. Such a system leveraging information from the textual parts of user reviews could then be used in scenarios like, for instance, a user looking for an Italian restaurant (cuisine type information captured from metadata) having very

courteous and friendly staff (sentiment towards the restaurant service can be captured from the text). Yet, there are several challenges in creating automatic aggregation and search tools that leverage the textual part of reviews. We discuss these challenges in automatic processing of user authored text in this chapter, and develop techniques to classify review text in Chapter 3.

2.1 Automatically Processing User Authored Text

User experience in accessing peer-authored content in reviews, forums and blogs can be enhanced if information on specific topics was automatically captured from the free-form textual content. A common approach to identifying topical information over text is to find the semantics associated with the individual terms. Utilizing existing taxonomies [75] like WordNet for computing such semantic coherence is very restrictive for capturing domain specific terms and their meaning. For instance, in the restaurant domain the text in user reviews contains several proper nouns of food dishes like Pho, Biryani or Nigiri, certain colloquial words like “apps” and “yum”, and certain words like “starter” which have clearly different meanings based on the domain (automobile reviews vs. restaurant reviews). WordNet will fail to capture such domain specific nuances.

Another well-studied approach for text analysis is clustering words based on their co-occurrences in the textual sentences [107, 24] and assigning a semantic class to each cluster. Yet, such clustering is not suitable for analyzing user reviews; reviews are typically small, and users often express opinions on several topics in the same sentence. For instance, in a restaurant reviews corpus our analysis showed that the words “food” and “service” which belong to obviously different restaurant aspects co-occur almost 10 times as often as the words “food” and “chicken”. On the contrary, utilizing the context around words can greatly assists in building topically coherent taxonomies.

An additional requirement for automatic review processing is identifying the sentiment of a statement. Sentiment detection is an open research question and is particularly challenging when used over user authored text. The same adjective can indicate a positive or a negative sentiment depending on which feature of a product is discussed, as in the following example:

Example 2: *The word “cheap” is polysemous in the restaurant domain.*

- *A satisfied reviewer about the restaurant Big Wong: "**Cheap eats** at a great price!"*
- *A dissatisfied reviewer about the restaurant Chow Bar: "The **decor was cheap looking** and the service was so-so."*

We address this challenge by capturing both the topical and sentiment information in unity from user generated text as described in Chapter 3. We can then disambiguate words that have a topic domain specific interpretation.

To complicate matters, some language constructs like sarcasm can confuse any automatic sentiment analysis tool, as in the following example:

Example 3: *The New York restaurant Pink Pony has 18 user reviews with an average star rating of 3, indicating that some reviews were positive while some were negative. This makes it further difficult to determine the sentiment of a sarcastic review, as shown here:*

- *"I had been searching really hard for a restaurant in New York where I could really feel unwanted and ignored and I finally found it! The staff ignored my friends and I the entire time we were there...**You guys are awesome!**"*

Sarcasm detection is a notably hard task [49]. Our techniques in Chapter 3 assess reviews at the sentence level, thus breaking down the effect of assigning oppositely polar sentiments to topics discussed earlier in the reviews. However, we are unable to detect sarcasm at the single sentence level, similar to a human-being who lacks context.

Finally, processing the genre of user reviews automatically differs from processing more traditional written texts, like news stories. The reviews contain unedited, often informal language. Poor spelling, unorthodox grammar and creative punctuation patterns introduce mistakes when using tools like parsers that have been trained on news stories.

2.2 Applications over User Authored Text

Due to the complexity of automatic processing of user authored text, this rich resource for information has been largely ignored while building applications like search and recommendations. Yet, textual data provides detailed opinions and experiences which are very valuable in

improving the user experience. Unfortunately, users often have to read through large amounts of text to find particular information of interest. We now discuss some additional challenges in utilizing user authored free-form text in enhancing search and making fine-grained rich recommendations.

2.2.1 Recommendations

Users should not have to read through several reviews, but should be presented with items that they would find interesting or useful based on some notion of preference through similarity with other users or items. This task of preference matching is carried out by recommendation systems [21]. Current recommendation systems such as the ones used by Netflix or Amazon [74] rely predominantly on structured metadata information to make recommendations, often using only the star ratings, and ignore a very important information source available in reviews: the textual content. Yet, utilizing text automatically in recommendation systems is challenging and requires identification of structure over free-form reviews.

First, the same individual words routinely appear in good and in bad reviews [10]. As such, a basic keyword-based search engine might not help users identify products with good reviews according to their information needs. Consider the following example:

Example 4: *The New York restaurant Bandol in* Citysearch *has several reviews discussing their desserts. However, these reviews often express opposite sentiments on the desserts as shown below, making it difficult to aggregate the opinions expressed by users:*

- *"Tiny dessert was $8.00...just plain overpriced for what it is."*
- *"The mussels were fantastic and so was the dessert ...definitely going to be back very soon."*

Another challenge is that a reviewer's overall rating might be largely reflective of product features in which the search user is not interested. Consider Example 1 from Chapter 1 where the New York restaurant Lucky Cheng's has 65 user reviews. Out of these, 40 reviews have a 4 or 5 star rating (out of 5 possible stars). Most of the positive reviews, however, focus on and praise the atmosphere of the restaurant. The negative reviews, on the other hand, complain

at length about the price and the service. A user not interested in atmosphere would not be interested in this restaurant.

Identifying which features of a product were discussed in a review automatically is a difficult task. Reviewers are creative in their writing, as shown in the example below:

Example 5: *The following sentences are all about the atmosphere and decor of restaurants, even though they share little in common, both in their content and style. Phrases like "feel more fast food than fine cuisine" expressing a negative sentiment or "Interior designers will be delighted" expressing a positive sentiment are difficult for computer systems to automatically identify:*

- *A reviewer writes about the restaurant Nadaman Hakubai: "Unflattering fluorescent lighting and less-than-luxurious furnishings make the space feel more fast food than fine cuisine [...]."*
- *A reviewer about the restaurant Tao: "Great music, beautiful people, great service..... except the part where you don't get seated right away even WITH a reservation."*
- *A reviewer about the restaurant Sea: "Interior designers will be delighted."*

Thus, capturing structure over user generated content in the form of topics and sentiments has many challenges, making it difficult to build automatic recommendation systems over user authored text. We describe our techniques to identify such useful information from the free-form text in Chapter 3.

2.2.2 Search

Users generate massive amounts of content daily in the form of reviews or forums. For instance, a popular forum on BMW cars sees as high as 50K unique visitors everyday. Such forums have a large amount of data; at the time of this analysis the BMW forum website had 25M posts and 0.6M members. A new BMW car owner is likely to have the same set of questions as a new owner from three months ago, i.e., users often have similar information needs. Such answers could be efficiently provided if good searching and ranking mechanisms are deployed over the large quantity of online user generated content.

Existing search mechanisms over online user generated content are very primitive; often posts containing query keywords are returned chronologically. As discussed above, keyword searched are not suitable as the same keywords routinely appear in both positive and negative reviews. Moreover, users are often interested in broad or vague features which cannot be directly captured by keywords. For instance, a user interested in "romantic" restaurants would like to find restaurants with soft music, candle-light dinner or Valentine's day specials.

User authored content often does not contain keywords pertaining to the item or feature searched. Background information is often omitted in posts as it is assumed that readers share the same background knowledge [34]. Consider the following example:

Example 6: *In a forum thread titled "Herceptin - Quick side effects poll" asking for side effects of a drug, several posts provide very relevant information without actually using the keywords "herceptin", "side" or "effects":*

- *"Congestion and constant sniffling for about a week. Also nose bleeds sometimes and persistent nose sores."*
- *"mild diarrhea for the first two days after"*
- *"Its a piece of cake compared to everything else I have been through. Yes to fatigue, runny nose and a little achy."*

A search mechanism retrieving posts alone will suffer from lack of context, and a user will have to read through the entire thread to understand which drug gives the above mentioned side effects.

A critical challenge then for web forums search is to provide results that are as complete as possible and that do not miss some relevant information but that are also focused on the part of individual threads or posts containing relevant text. Therefore, in this type of search the correct level of result granularity is important. When issuing search queries to forums, users face the daunting task of sifting through a massive number of posts to separate the wheat from the chaff.

Example 7: *Consider the search results in Example 7 retrieved in response to the user query hair loss in a breast cancer patient forum. Several succinct sentences (A) through (H), shown*

in boldface, are highly relevant to the query and provide very useful answers. Yet, when a post contains many relevant sentences as in Post1 and Post2, the post is a better result than the sentences alone.

- Post1: **(A) Aromasin certainly caused my hair loss and the hair started falling 14 days after the chemo.** *However, I bought myself a rather fashionable scarf to hide the baldness. I wear it everyday, even at home.* **(B) Onc was shocked by my hair loss so I guess it is unusual on Aromasin.** *I had no other side effects from Aromasin, no hot flashes, no stomach aches or muscle pains, no headaches or nausea and none of the chemo brain.*
- Post2: **(C) Probably everyone is sick of the hair loss questions, but I need help with this falling hair.** I had my first cemotherapy on 16th September, so due in one week for the 2nd treatment. **(D) Surely the hair loss can't be starting this fast..or can it?.** *I was running my fingers at the nape of my neck and about five came out in my fingers. Would love to hear from anyone else have AC done (Doxorubicin and Cyclophosphamide) only as I am not due to have the 3rd drug (whatever that is - 12 weekly sessions) after the 4 sessions of AC. Doctor said that different people have different side effects, so I wanted to know what you all went through.* **(E) Have n't noticed hair loss elsewhere, just the top hair and mainly at the back of my neck. (F) I thought the hair would start thining out between 2nd and 3rd treatment, not weeks after the 1st one.** *I have very curly long ringlets past my shoulders and am wondering if it would be better to just cut it short or completely shave it off. I am willing to try anything to make this stop, does anyone have a good recommendation for a shampoo, vitamins or supplements and (sadly) a good wig shop in downtown LA.*
- Post3: *My suggestion is, don't focus so much on organic. Things can be organic and very unhealthy. I believe it when I read that nothing here is truly organic. They're allowed a certain percentage. I think 5% of the food can not be organic and it still can carry the organic label. What you want is nonprocessed, traditional foods. Food that comes from a farm or a farmer's market. Small farmers are not organic just because it is too much trouble to get the certification. Their produce is probably better than most of the industrial organic stuff.* **(G) Sorry Jennifer, chemotherapy and treatment followed**

by hair loss is extremely depressing and you cannot prepare enough for falling hair, especially hair in clumps. (H) I am on femara and hair loss is non-stop, I had full head of thick hair.

Dynamically selecting the best level of focus on the data helps users find relevant answers without having to read large amounts of irrelevant text.

Therefore, our goal is to improve the experience of users searching through web forums by providing results that focus on the parts of the data that best fit their information needs. Our approach in Chapter 5 allows for search results to be returned at varying granularity levels: single pertinent sentences containing facts, larger passages of descriptive text, or entire discussions relevant to the search topics.

2.3 Personalization Challenges

Search and recommendations over user authored content can greatly benefit from personalization of results returned to a user. Online users share many similarities and finding like-minded users enables us to tailor the user experience to their specific preferences and needs.

An under utilized signal in forum data is the information on authorship of posts. Since most online forums require contributors to register, forum participants have an identity and one can leverage the inherent social interactions between forum participants to enhance search. User interactions provide vital clues about their information needs, topics of interest and their preferred other forum participants to answer questions. Some users are very prolific and knowledgeable, and participate in many different discussions on varying topics. Such users are likely to contribute high quality information to forums and their content should have a higher ranking score. Alternately, some users share the same information need, are similar to each other and can benefit from each other. Consider for instance, the study in [57] shows that patients of a particular stage of cancer (Stage I through IV) are more likely to interact with other users with the same progression of the disease. Finding such similar users and weighting their content strongly will enhance the personalized experience of a user. Unfortunately, online forums largely do not provide such personalized search functionality.

Finding similarities in forum participants can enable a search system to retrieve more useful and relevant results authored by like-minded users. Finding such personalized similarity is a complex problem due to the mix of various signals of user affinity. Occasionally, web forums allow users to make explicit friendships, thus providing the social graph over users. Additionally, there exist several implicit cues of user affinity like participating in the same threads or discussing the same topics. Yet, two users having similar information needs at different times might never participate in the same discussions. For instance, in a forum for mothers several participants will have similar questions about feeding, teething, and sleep patterns. However, some mothers with older children will never participate in newer threads related to infants, even though they have experience raising infants and may have participated in such threads in the past. On the other hand, for a location-based business search, forum participants in the query location are likely to provide answers despite largely varying profiles or topics of interest. Therefore, it is an important and challenging problem to uniformly capture similarities in online users while taking into account multiple implicit signals like user profiles, topics of interest or information needs and assigning egocentric importance to each of these interpersonal relations. We discuss our approach to address the problem of finding like-minded forum participants is to find a multidimensional user similarity score that takes into account the egocentric importance associated by a user to the different interpersonal relations, described in detail in Section 6.

Thus, extracting relevant features from text and discovering user sentiment towards these features is a challenging task, and utilizing this free-form user authored text in search and recommendation systems poses several interesting challenges. In the following chapter, we discuss our techniques for automatically processing user generated text to capture important topics and user sentiment expressed towards these specific topics. Such a fine-grained textual analysis helps in addressing several challenges discussed in this chapter.

Chapter 3

Structure Identification over Text

Web reviews have a combination of linguistic characteristics that depart from the genres traditionally considered in the field of information processing: the language is often quite specific to a particular domain (reviewers of electronic goods, for instance, use many technical terms to describe product features like resolution, battery life, zoom); at the same time reviews are unedited and often contain informal and ungrammatical language. Certain language constructs like sarcasm, make it difficult to identify review sentiment using words as indicators, as described in the previous chapter. Finally, reviews often contain anecdotal information, which does not provide useful, or usable, information for the sake of automatic processing.

For automatic processing of user authored text in applications like search and recommendations, we develop techniques to identify structure over this free-form text in the form of domain-specific topics and user sentiment towards these features. In this chapter, we detail our methods for supervised classification of text at the sentence level in Section 3.1. While supervised classification provides very accurate topical and sentiment analysis, such techniques are expensive due to the cost of manual annotation of labeled samples. In Section 3.2, we develop a semi-supervised method for discovering the most meaningful words representing the topics of interest, and show that we propagate these topical labels nearly optimally over the text graph.

3.1 Supervised Classification of Review Sentences

Our approach to addressing most of the challenges from Chapter 2 is to consider a review not as a unit of text, but as a set of sentences each with their own topics and sentiments. This added structural information provides valuable information on the textual content at a fine-grain level. We model our approach as a multi-label text classification task for each sentence where labels are both about topics and sentiments. We focused our classification effort on a restaurant review

data set, described in Section 3.1.1. We report on our classification effort in Section 3.1.2, and on the results of our analysis of user reviewing patterns in Section 3.1.4.

3.1.1 Data Set

We focused our classification effort on a restaurant review data set, extracted from the NY Citysearch web site [1]. The corpus contains 5531 restaurants, with associated structured information (location, cuisine type) and a set of reviews. There are a total of 52264 reviews. Reviews contain structured metadata (star rating, date) along with text. Typically reviews are small; the average user review has 5.28 sentences. The reviews are written by 31814 distinct users, for whom we only have unique username information.

The data set is sparse: restaurants typically have only a few reviews, with 1388 restaurants having more than 10 reviews; and users typically review few restaurants, with only 299 (non-editorial) users having reviewed more than 10 restaurants.

3.1.2 Structure Identification and Analysis

As the first step, we analyzed the data to identify categories specific to the restaurant reviews domain. These dimensions focus on particular aspects of a restaurant. We identified the following six categories: Food, Service, Price, Ambience, Anecdotes, and Miscellaneous. The first four categories are typical parameters of restaurant reviews (e.g., Zagat ratings). Anecdotal sentences are sentences describing the reviewer's personal experience or context, but that do not usually provide information on the restaurant quality (e.g. *"I knew upon visiting NYC that I wanted to try an orginal deli"*). The Miscellaneous category captures sentences that do not belong to the other five categories and includes sentences that are general recommendations (e.g., *"Your friends will thank you for introducing them to this gem!"*). Sentence categories are not mutually exclusive and overlap is allowed.

In addition to sentence categories, sentences have an associated sentiment: Positive, Negative, Neutral, or Conflict. Users often seem to compare and contrast good and bad aspects; this

[1] `Classified and original data can be downloaded at http://www.research.rutgers.edu/~gganu/datasets`

Sentence Category	Accuracy	Precision	Recall
FOOD	84.32	81.43	76.72
SERVICE	91.92	81.00	72.94
PRICE	95.52	79.11	73.55
AMBIENCE	90.99	70.10	54.64
ANECDOTES	87.20	49.15	44.26
MISCELLANEOUS	79.40	61.28	64.20
Sentiment	**Accuracy**	**Precision**	**Recall**
POSITIVE	73.32	74.94	76.60
NEGATIVE	79.42	53.23	45.68
NEUTRAL	80.86	32.34	23.54
CONFLICT	92.06	43.96	35.68

Table 3.1: 7-Fold cross validation of classifier results.

mixed sentiment is captured by the Conflict category (e.g., *"The food here is rather good , but only if you like to wait for it"*).

Manual Sentence Annotation

To classify sentences into the above mentioned categories and sentiment classes, we manually annotated a training set of approximately 3400 sentences with both category and sentiment information. To check for agreement, 450 of these sentences were annotated by three different annotators. The kappa coefficient measures pairwise agreement among a set of annotators making category judgments, correcting for expected chance agreement [91]. A Kappa value of 1 implies perfect agreement, the lower the value, the lower the agreement. The inter-annotator agreements for our annotations were very good (Kappa above 0.8) for the Food, Price, and Service categories and Positive sentiment. The Negative sentiment (0.78), Neutral and Conflict sentiments, Miscellaneous and Ambience categories all had good agreements (above 0.6). The ambiguous Anecdotes category is the only one for which the Kappa value was moderate (0.51).

Automatic Sentence Classification

We trained and tested Support Vector Machine classifiers [58] on our manually annotated data (one classifier for each topic and one for each sentiment type). Features for all classifiers were stemmed words (extensive experiments did not suggest significant improvements in accuracy when other features like n-grams and parts-of-speech tags were used for classification). We

used svm light[2] with default parameters for building our text classifiers.

We performed 7-fold cross validation [65] and used accuracy, precision and recall to evaluate the quality of our classification (see Table 3.1). Precision and recall for the main categories of Food, Service and Price and the Positive sentiment were high (70%), while they were lower for the Anecdotes, Miscellaneous, Neutral and Conflict categories. These low results could be due to the ambiguous nature of these categories but also due to the small amount of training instances in our corpus for these categories in particular.

While the specific categories we identified are tailored for a restaurant scenario, our classification approach could easily be translated to other types of data sets after a topical analysis to identify product-specific sentence categories.

3.1.3 Cuisine-specific Classification

As described in EXAMPLE 4 in Chapter 2, some words have different meanings in different contexts and could confuse both the topic and sentiment classification. Our corpus contains metadata tags like cuisine type or location that can be used for disambiguation. To verify this hypothesis we conducted the following experiment: we controlled for the cuisine type in both the training and testing sets. Sentences of a particular cuisine were used to train a classifier, and the classifier was tested with sentences belonging to the same cuisine. These classifiers yield significantly more accurate results than in the general case (see Figure 3.1). This result confirms our intuition that metadata tags can be used to guide text classifiers. Since the gold standard does not span all the cuisines in the dataset with sufficient representative sentences (only 10 cuisine types in the corpus were included in the gold standard, and the results in Figure 3.1 are observed using 5 cuisine types with an average of 268 sentences of each cuisine type), we do not investigate this type of stratification further. We acknowledge, however, the potential improvement in classification accuracy.

---

[2] `http://svmlight.joachims.org`

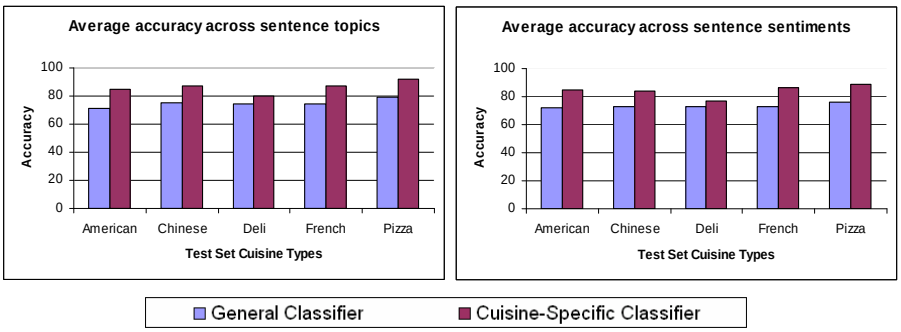

Figure 3.1: Comparison between accuracy of general and cuisine specific classification.

3.1.4 Text Review Analysis

To understand trends in reviewing behaviors, we performed an in-depth analysis of the corpus of 52264 user reviews augmented with our automatic classification. Thus, we could study the relation between the textual structure of the reviews and the metadata entered by the reviewers, such as star rating. While we uncovered some surprising and interesting trends, we also confirmed some obvious and expected behavior. The later shows that our classification is sound and our analysis of textual information conforms to the expected behaviors in user reviews.

User Reviewing Trends

Our analysis of the annotated corpus of reviews shows that the sentiment expressed in the reviews was mostly positive (56% of sentences), while only 18% of the review sentences expressed negative sentiment. This is consistent with the star ratings provided by users, with 73% of reviews having a star rating of 4 or 5.

Most reviews describe the food served by the restaurant (32%), while fewer than 17% of the sentences are about the service, 10% are about ambience and 6.5% are about price. The distribution of topics covered in the reviews is dependent on the cuisine type (metadata) of the restaurant as discussed below. The distribution of sentence categories and sentiments are shown in Figure 3.2.

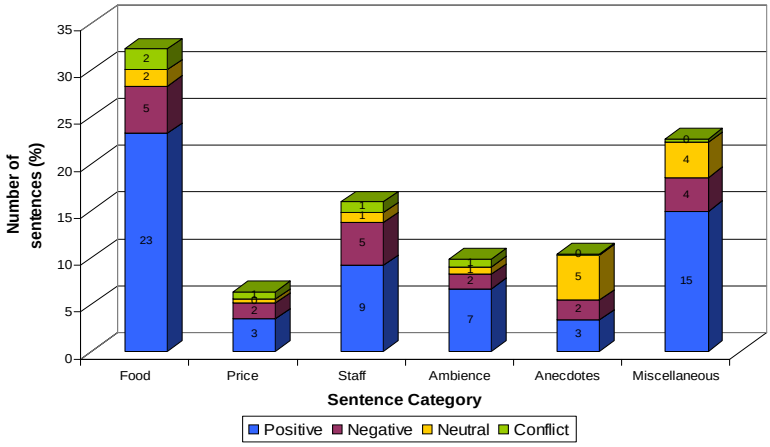

Figure 3.2: Distribution of the categories and sentiments of automatically classified sentences.

Dependence on Cuisine Type

We observe that in addition to the free-form textual data in reviews there is valuable information in the structured metadata associated with the reviews, and in fact these two parts of the reviews are often related. We now explore the correlation between the unstructured text and the structured metadata.

The distribution of topics in reviews is dependent on the cuisine classification (metadata) of the restaurant. Restaurants serving French and Italian cuisines have many Service related sentences (20%). Surprisingly most of these sentences for French restaurants were Negative (50%) while for Italian restaurants these sentences were mostly Positive (72%). In contrast, reviews of Chinese restaurants, Delis and Pizzerias focus mostly on Food.

Dependence on Price-Level

Coarse price level metadata information (numerical value from 1 to 4, 1 being the cheapest) is associated with restaurants in the data set. Figure 3.3 shows that the number of positive price related sentences decreases and the number of negative price related sentences increases as the metadata price level increases implying, unsurprisingly, that users complain more about prices of expensive restaurant.

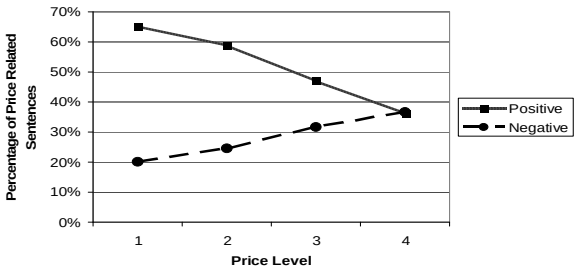

Figure 3.3: Impact of price level on perception.

Comparing Star Rating with Sentiment

Probably the most important metadata information in reviews is the user-input star rating (from 1 to 5 in our data set, 5 being the highest). We compare this star rating with the sentiment annotation produced by our classifier using the Pearson correlation coefficient [89]. The coefficient ranges from -1 to 1, with -1 for negative correlation, 1 for positive correlation and 0 for no correlation. Our results show a positive correlation (0.45) between the star rating and the percentage of positive sentences in the review, and a negative correlation (-0.48) between the star rating and the percentage of negative sentences. On average, reviews with good ratings of 4 and 5 mainly have positive sentences (71%), and very few negative sentences (6%). In contrast, reviews with bad star ratings (1 or 2) have 5% positive sentences and above 78% negative sentences. These observations and the much finer range of interpretations of text reviews gives us motivation to include text in a restaurant recommendation system, as described in Chapter 4.

While supervised classification suffers from requiring a large quantity of labeled instances, our experiments indicated that this technique yielded more accurate results. We use the topical and sentiment information captured at the sentence level in Chapter 4 for building text-based recommendation systems. Next, we describe an alternate semi-supervised method for topical analysis [44] that can be easily ported to new domains or new definitions of features to be discovered.

3.2 Semi Supervised Text Classification

Semi-supervised learning is a field of machine learning that studies learning from both labeled and unlabeled examples. In practice, large amounts of data are available but only a tiny fraction

of these data is labeled. Such problems can be often cast as semi-supervised learning problems and various methods for solving these problems exist [109]. The focus of this section is graph-based learning [110], and in particular the harmonic solution (HS) on graphs, which serves as a basis for many semi-supervised learning algorithms [15, 109].

In this section, we show how the harmonic solution on a graph can be approximated by the HS on its subgraph, with provable guarantees on the quality of the approximation. The premise of our technique is that the subgraph is much smaller than the graph and thus the HS on the subgraph can be computed more efficiently. We state conditions under which the subgraph yields a good approximation, prove bounds on the quality of the approximation, and show how to build the graph efficiently. Our method is evaluated on handwritten digit recognition and two bigger problems: topic discovery in restaurant and hotel reviews. Our experimental results support our claims that only a small portion of the graph is usually needed to identify topics on the entire graph.

The techniques in this section as discussed in [44] address an important problem. In practice, data are large-scale and often only a small portion of labels can be inferred with high confidence. The low confidence predictions are typically of little utility because they are noisy. We show how to identify high confidence predictions without the overhead of modeling the low confidence ones. This is the first such result for graph-based semi-supervised learning.

Our algorithm for building subgraphs (Section 3.2.1) can be viewed as a form of self-training. Similarly to self-training, the algorithm is iterative and easy to implement. Unlike self-training, we provide guarantees on the quality of the solution. In other words, we show how to do self-training in a proper way.

This section is organized as follows. First, we introduce basic concepts, such as semi-supervised learning on graphs (Section 3.2.1) and self-training (Section 3.2.1). Second, we motivate our approach, analyze the error in the estimate of the harmonic solution on the subgraph, and propose a method for building the subgraph (Section 3.2.2). Finally, we evaluate the method on three datasets, two of which are large-scale (Sections 3.2.3 and 3.2.4).

3.2.1 Semi-supervised learning: Preliminaries

We adopt the following notation. The symbols $\mathbf{x}_i$ and y_i refer to the i-th example and its label, respectively. All examples belong to one of c classes $y \in \{1, \ldots, c\}$. The examples are divided into the labeled and unlabeled sets, l and u, and we only know labels in the labeled set l. The cardinality of these sets are n_l and n_u. The total number of the examples is $n = n_l + n_u$.

Semi-supervised learning can be formulated as label propagation on a graph, where the vertices are the examples $\mathbf{x}_i$ [110]. The labels can be computed by solving a system of linear questions:

$$F_u = -(L_{uu})^{-1} L_{ul} F_l, \tag{3.1}$$

where $F \in \{f_i[k]\}^{n \times c}$ is a matrix of probabilities that the vertex i belongs to the class k; F_l and F_u are the rows of F corresponding to the labeled and unlabeled vertices, respectively; $L = D - W$ is the *Laplacian* of the data adjacency graph W; and D is a diagonal matrix whose i-th diagonal entry is computed as $d_i = \sum_j w_{ij}$. We refer to the i-th row of F as $\mathbf{f}_i = (f_i[1], \ldots, f_i[c])$ and to the *most confident prediction* in the row as $\|\mathbf{f}_i\|_\infty = \max_k |f_i[k]|$. We adopt the convention that vertices are indexed by i and j, and labels by k.

The solution F_u is known as the *harmonic solution (HS)* because it satisfies the *harmonic property* $f_i[k] = \frac{1}{d_i} \sum_{j \sim i} w_{ij} f_j[k]$. It can be rewritten as:

$$\begin{aligned} F_u &= (I - P_{uu})^{-1} P_{ul} F_l \\ &= (I + P_{uu} + P_{uu}^2 + \ldots) P_{ul} F_l, \end{aligned} \tag{3.2}$$

where $P = D^{-1} W$ is a transition matrix of a random walk on W. As a result, $f_i[k]$ can be viewed as the probability that the random walk that starts from the vertex i is absorbed at vertices with labels k [110]. Our work relies heavily on this interpretation.

Large-scale semi-supervised learning

In general, the space and time complexity of computing the harmonic solution (Equation 3.1) are $\theta(n^2)$ and $\theta(n^3)$, respectively. So it is challenging to compute the exact solution when the number of vertices n exceeds several thousand. The computation can be sped up significantly by taking into account the structure of the problem. For instance, when the graph W is $O(n)$

sparse, the time complexity of computing the HS is $\theta(n^2)$ [109]. Moreover, when the system of linear equations (Equation 3.1) is properly preconditioned [95], it can be solved approximately in nearly linear time in n. Data dependent kernels for semi-supervised max-margin learning can be also built in nearly linear time [71].

A few methods can approximate the HS in $\theta(n)$ time [98, 38, 102]. Fergus *et al.* [38] cast this problem as regression on basis functions, which are eigenfunctions over features. Valko *et al.* [102] choose k representative vertices, compute the HS on these vertices, and then propagate the solution to the rest of the graph. The space and time complexity of this method are $\theta(k^2)$ and $\theta(kn + k^3)$, respectively. Our solution is later compared to this approach.

Our work is orthogonal to the existing work on large-scale semi-supervised learning on graphs. In particular, we study the problem where only a small portion of the graph, typically in the vicinity of labeled vertices, is sufficient to identify most confident predictions and show how this subgraph can be learned efficiently. Our method can be easily combined with existing approximations, such as data quantization [102]. In this case, we would learn an approximation to the subgraph.

Self-training

Self-training [106] is one of the oldest and most popular methods for semi-supervised learning. In self-training, a classifier is initially trained on a few labeled examples. Then it is used to predict labels of unlabeled examples, the most confident predictions are added to the training set, the classifier is retrained, and this is repeated until a stopping criterion is met. Self-training is very common in natural language processing, and was applied to various problems, such as named-entity [35, 82] and relation [22, 7, 83] extraction.

The disadvantage of self-training is that it is subject to local optima and does not provide guarantees on the quality of the approximation [110]. Our algorithm for learning ε-subgraphs (Algorithm 1) can be viewed as a type of self-training. Similarly to self-training, the method is iterative and easy to implement. Unlike self-training, we provide guarantees on the quality of the solution.

3.2.2 Inference on a Subgraph

We now introduce our setting and demonstrate subgraph inference on two example problems. Next, we identify a class of subgraphs called ε-subgraphs, and bound the quality of the HS approximations for these graphs. Finally, we propose a technique for constructing ε-subgraphs and discuss how to apply it in practice.

Subgraphs

We want to efficiently approximate the harmonic solution on the graph by the HS on its subgraph, defined as follows:

Definition 1. *A **subgraph** $W[e]$ of a graph W induced by vertices e is a graph over e where two vertices are adjacent if and only if they are adjacent in W. The subgraph $W[e]$ is given by a $n \times n$ adjacency matrix:*

$$(W[e])_{ij} = \begin{cases} W_{ij} & (i \in e) \wedge (j \in e) \\ 0 & \text{otherwise.} \end{cases}$$

In other words, the edge W_{ij} appears in the subgraph $W[e]$ only if both i and j are subgraph vertices e.

We refer to the harmonic solutions on W and its subgraph $W[e]$ as F^* and F^e, respectively. The respective solutions at the vertex i are denoted by $\mathbf{f}_i^*$ and $\mathbf{f}_i^e$. If the vertex $i \notin e$ is not in the subgraph, we assume that $\mathbf{f}_i^e = \mathbf{0}$.

Our goal is to learn a subgraph such that the difference between the harmonic solutions $\mathbf{f}_i^*$ and $\mathbf{f}_i^e$ is bounded at vertices $i \in e$. The difference is measured by the *max-norm distance*:

$$\|\mathbf{f}_i^e - \mathbf{f}_i^*\|_\infty = \max_k |f_i^e[k] - f_i^*[k]| \,. \tag{3.3}$$

We opt for this distance because it allows us to identify highly confident predictions on W. In particular, note that when $\|\mathbf{f}_i^e - \mathbf{f}_i^*\|_\infty$ is small and the confidence $f_i^e[k]$ on the subgraph is

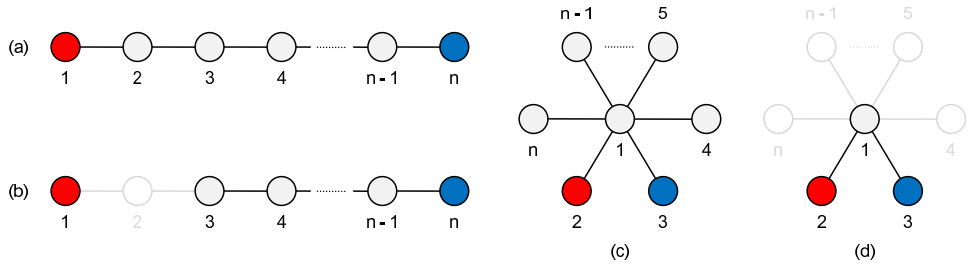

Figure 3.4: **a.** A *line graph* of n vertices. **b.** A subgraph of the line graph on vertices $e = \{1, 2, 3, \ldots, n\}$. **c.** A *star graph* of n vertices. **d.** A subgraph of the star graph on vertices $e = \{1, 2, 3\}$.

high, then the corresponding confidence $f_i^*[k]$ on W is bounded from below as:

$$\begin{aligned} f_i^*[k] &= f_i^e[k] - (f_i^e[k] - f_i^*[k]) \\ &\geq f_i^e[k] - \max_k |f_i^e[k] - f_i^*[k]| \\ &= f_i^e[k] - \|\mathbf{f}_i^e - \mathbf{f}_i^*\|_\infty , \end{aligned} \tag{3.4}$$

and is also high. In the rest of the section, we discuss two examples of inference on subgraphs.

The first example is a *line graph* of n vertices (Figure 3.4a):

$$W_{ij} = \mathbb{1}\{|j - i| = 1\} \quad \forall\, i, j \in \{1, \ldots, n\} . \tag{3.5}$$

In this graph, all consecutive vertices, i and $i + 1$, are adjacent. The vertices 1 and n are labeled as classes 1 and 2, respectively. Let the subgraph $W[e]$ be induced by $n - 1$ vertices $e = \{1, 3, 4, \ldots, n\}$ (Figure 3.4b), all but the vertex 2. Moreover, let the size of the graph n increase. Then note that $\mathbf{f}_3^* \to (1, 0)$ because the distance of the vertex 3 to labeled vertices 1 and n remains constant and increases linearly with n, respectively. On the other hand, $\mathbf{f}_3^e = (0, 1)$ for all n because the labeled vertex 1 cannot be reached from the vertex 3. So the distance between the harmonic solutions $\|\mathbf{f}_3^e - \mathbf{f}_3^*\|_\infty \to 1$. In other words, the HS on subgraphs can be inaccurate, even when the subgraph covers a large portion of the entire graph W.

The second example is a *star graph* of n vertices (Figure 3.4c):

$$W_{ij} = \mathbb{1}\{(i = 1) \vee (j = 1)\} \quad \forall\, i, j \in \{1, \ldots, n\} . \tag{3.6}$$

In this graph, all vertices are adjacent to a single central vertex. The vertices 2 and 3 are labeled as classes 1 and 2, respectively. Let the subgraph $W[e]$ be induced by 3 vertices $e = \{1, 2, 3\}$

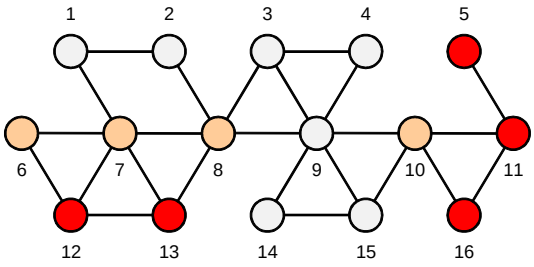

Figure 3.5: An ε-subgraph over nine vertices. The dark and light red vertices are the core and surface vertices of the ε-subgraph, respectively. The subgraph edges are shown in red color.

(Figure 3.4d). Moreover, let the size of the graph n increase. Note that regardless of n, $\mathbf{f}_1^* = \mathbf{f}_1^e = (\frac{1}{2}, \frac{1}{2})$. So the distance between the two solutions $\|\mathbf{f}_1^e - \mathbf{f}_1^*\|_\infty$ is zero for all n. In other words, the HS on subgraphs can be highly accurate, even when the subgraph covers only a tiny portion of the entire graph W.

ε-subgraphs

Our examples illustrate that the HS on a subgraph can be both a good and bad approximation of that on the entire graph. The quality of the approximation depends on how the subgraph is constructed. In this work, we analyze a special family of subgraphs that provide guarantees on the quality of the approximation.

Definition 2. *An ε-**subgraph** $W[e]$ induced by vertices e is a subgraph that has two additional properties. First, the set e can be divided into* core*:*

$$e^+ = \{i \in e : \|\mathbf{f}_i^e\|_\infty > \varepsilon\}$$

surface*:*

$$e^- = \{i \in e : \|\mathbf{f}_i^e\|_\infty \leq \varepsilon\}$$

vertices such that all neighbors of the core vertices e^+ in W are in e. Second, all labeled vertices l are in e.

Informally, we refer to the vertices whose most probable label is predicted with at least ε probability, $\|\mathbf{f}_i\|_\infty > \varepsilon$, and no more than ε probability, $\|\mathbf{f}_i\|_\infty \leq \varepsilon$, as *high* and *low confidence* predictions, respectively. The core e^+ and surface e^- vertices in the ε-subgraph are examples of high and low confidence predictions, respectively.

The ε-subgraph is a graph where high confidence predictions are separated from the vertices $u \setminus e$ that are not in the subgraph by the low confidence ones (Figure 3.5). Due to this structure, we can make two claims. First, if a vertex is not predicted with high confidence on the subgraph, it cannot be predicted with high confidence on the graph. Second, if a prediction on the subgraph is highly confident, it is also confident on the graph.

In the rest of this section, we prove these claims. First, we show that if a vertex is not a core vertex, it cannot be predicted with high confidence on W.

Proposition 1. *Let $i \in u \setminus e^+$. Then $\|\mathbf{f}_i^*\|_\infty \leq \varepsilon$.*

Proof: Our proof is based on the random-walk interpretation of the HS. In particular, $f_i^*[k]$ is the probability that random walks on W that start in the vertex i are absorbed at vertices with labels k. Note that $W[e]$ is an ε-subgraph. Therefore, $l \subseteq e^+$ and all neighbors of e^+ are in e. So all random walks on W from $i \in u \setminus e^+$ must visit at least one surface vertex before being absorbed. Let j be the first such vertex after no vertex from $u \setminus e$ is visited. Then $f_i^*[k]$ can be rewritten as:

$$f_i^*[k] = \sum_{j \in e^-} \phi_{ij}^k f_j^e[k], \tag{3.7}$$

where ϕ_{ij}^k is the fraction of the aforementioned random walks that correspond to the vertex j and $f_j^e[k]$ is the absorption probability at this vertex. Note that $f_j^e[k]$ is bounded from above by ε. Therefore, it follows:

$$\|\mathbf{f}_i^*\|_\infty \leq \max_k |f_i^e[k]| \leq \max_k \Big| \underbrace{\sum_{j \in e^-} \phi_{ij}^k}_{1} f_j^e[k] \Big| \leq \varepsilon. \tag{3.8}$$

This concludes our proof. ■

Proposition 1 says that if a vertex is not a core vertex, it can only be predicted with low confidence on W. Another way of interpreting the result is that only the core vertices e^+

can be ever predicted with high confidence. In the following claim, we bound the difference between the harmonic solutions on W and its subgraph $W[e]$ on the core vertices e^+.

Proposition 2. *Let $i \in e^+$. Then:*

$$\|\mathbf{f}_i^e - \mathbf{f}_i^*\|_\infty \leq \frac{1 - \|\mathbf{f}_i^e\|_\infty}{1 - \varepsilon}.$$

Proof: Our proof consists of two main steps. First, we bound from below the fraction of random walks that visit only the core vertices e^+. Since all neighbors of the vertices e^+ are in $W[e]$, these walks do not change between W and $W[e]$. Based on this fact, we prove an upper bound on the difference in the harmonic solutions.

Let p_{ik}^+ and p_{ik}^- be probabilities that random walks on $W[e]$ that start in the vertex i, and visit only the core vertices and at least one surface vertex, respectively, are absorbed at vertices with labels k. Then the probability that the vertex i has label k can be written as:

$$f_i^e[k] = \alpha p_{ik}^+ + (1 - \alpha) p_{ik}^-, \tag{3.9}$$

where α is the fraction of the walks that visit only the core vertices e^+. The fraction α can be bounded from below as:

$$\alpha = \frac{f_i^e[k] - p_{ik}^-}{p_{ik}^+ - p_{ik}^-} \geq \frac{f_i^e[k] - p_{ik}^-}{1 - p_{ik}^-} \geq \frac{f_i^e[k] - \varepsilon}{1 - \varepsilon}. \tag{3.10}$$

The last inequality is due to the fact that $\frac{f_i^e[k] - p_{ik}^-}{1 - p_{ik}^-}$ decreases when p_{ik}^- increases and $f_i^e[k] \leq 1$. We bound p_{ik}^- from above by ε. This upper bound follows from the observation that the probability p_{ik}^- can be rewritten as:

$$p_{ik}^- = \sum_{j \in e^-} \phi_{ij}^k f_j^e[k], \tag{3.11}$$

where ϕ_{ij}^k is the fraction of random walks from the vertex i where the first visited surface vertex is j and $f_j^e[k]$ is the absorption probability at the vertex j. Since $W[e]$ is an ε-subgraph, the probability $f_j^e[k]$ can be bounded from above by ε. So any convex combination of $f_j^e[k]$, such as p_{ik}^-, is bounded by ε.

The lower bound on the fraction α (Equation 3.10) holds for all k. So we bound α from below by the largest one with respect to k:

$$\alpha \geq \max_k \left\{ \frac{f_i^e[k] - \varepsilon}{1 - \varepsilon} \right\} = \frac{\|\mathbf{f}_i^e\|_\infty - \varepsilon}{1 - \varepsilon}. \tag{3.12}$$

Since the fraction of random walks that visit only the core vertices e^+ is bounded from below, and these walks are the same on W and $W[e]$, the difference in the harmonic solutions on W and $W[e]$ can be bounded from above as:

$$|f_i^e[k] - f_i^*[k]| \leq 1 - \frac{\|\mathbf{f}_i\|_\infty - \varepsilon}{1-\varepsilon} = \frac{1 - \|\mathbf{f}_i\|_\infty}{1-\varepsilon}. \tag{3.13}$$

This concludes our proof. ■

Proposition 2 says that highly-confident predictions on the subgraph $W[e]$ are good approximations of those on the graph W. For instance, when $f_i^e[k] = 0.9$ and $\varepsilon = 0.5$, the distance $\|\mathbf{f}_i^e - \mathbf{f}_i^*\|_\infty$ cannot be larger than 0.2, no matter how much W and $W[e]$ differ. Our upper bound on $\|\mathbf{f}_i^e - \mathbf{f}_i^*\|_\infty$ can be applied to Equation 3.4 and yields the following lower bound.

Proposition 3. *Let $i \in e^+$. Then:*

$$f_i^*[k] \geq f_i^e[k] - \frac{1 - \|\mathbf{f}_i^e\|_\infty}{1-\varepsilon}.$$

Normalized Laplacian

The Laplacian L in the harmonic solution is often substituted for the *normalized Laplacian* $L_n = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. In this section, we show how to generalize our results to the normalized Laplacian L_n. The main difficulty is that the HS on L_n does not have a clear random-walk interpretation. To obtain it, we rewrite the HS as:

$$\begin{aligned} F_u &= (I - D_{uu}^{-\frac{1}{2}} W_{uu} D_{uu}^{-\frac{1}{2}})^{-1} D_{uu}^{-\frac{1}{2}} W_{ul} D_{ll}^{-\frac{1}{2}} F_l \\ &= (D_{uu}^{\frac{1}{2}} (I - D_{uu}^{-1} W_{uu}) D_{uu}^{-\frac{1}{2}})^{-1} D_{uu}^{-\frac{1}{2}} W_{ul} D_{ll}^{-\frac{1}{2}} F_l \\ &= D_{uu}^{\frac{1}{2}} (I - D_{uu}^{-1} W_{uu})^{-1} D_{uu}^{-1} W_{ul} D_{ll}^{-\frac{1}{2}} F_l \\ &= D_{uu}^{\frac{1}{2}} (I - P_{uu})^{-1} P_{ul} D_{ll}^{-\frac{1}{2}} F_l. \end{aligned} \tag{3.14}$$

Finally, we multiply both sides of Equation 3.14 by $D_{uu}^{-\frac{1}{2}}$ and get:

$$\begin{aligned} [D_{uu}^{-\frac{1}{2}} F_u] &= (I - P_{uu})^{-1} P_{ul} [D_{ll}^{-\frac{1}{2}} F_l] \\ &= (I + P_{uu} + P_{uu}^2 + \dots) P_{ul} [D_{ll}^{-\frac{1}{2}} F_l]. \end{aligned} \tag{3.15}$$

Equation 3.15 has the same form as Equation 3.2. As a result, we may conclude that the HS on the normalized Laplacian L_n has indeed a random walk interpretation, where the HS F_u is additionally scaled by $D_{uu}^{-\frac{1}{2}}$.

Let the self-similarity w_{ii} of all vertices i be 1. Then $d_i \geq 1$ for all vertices i. Moreover, note that $||\mathbf{f}_i||_1 = 1$ at all labeled vertices because $\mathbf{f}_i$ is a distribution over labels. As a result, $||d_i^{-\frac{1}{2}}\mathbf{f}_i||_1 \leq 1$ at these vertices. From Equation 3.15, it follows that $||d_i^{-\frac{1}{2}}\mathbf{f}_i||_1 \leq 1$ for all i. So $d_i^{-\frac{1}{2}}\mathbf{f}_i$ can be loosely interpreted as a distribution.

Based on our discussion, the HS on the normalized Laplacian L_n has a random-walk interpretation (Equation 3.15), and $||d_i^{-\frac{1}{2}}\mathbf{f}_i||_\infty \leq ||d_i^{-\frac{1}{2}}\mathbf{f}_i||_1 \leq 1$ at all vertices i. Therefore, we can follow the same reasoning as in Section 3.2.2 and generalize our results as follows.

Proposition 4. *Let F^* and F^e be harmonic solutions on the normalized Laplacians of W and its ε-subgraph $W[e]$. Let the* core *and* surface *vertices be defined as:*

$$e_n^+ = \left\{ i \in e : d_i^{-\frac{1}{2}} \left\|\mathbf{f}_i^e\right\|_\infty > \varepsilon \right\}$$

and

$$e_n^- = \left\{ i \in e : d_i^{-\frac{1}{2}} \left\|\mathbf{f}_i^e\right\|_\infty \leq \varepsilon \right\},$$

respectively. Then:

$$d_i^{-\frac{1}{2}} \left\|\mathbf{f}_i^*\right\|_\infty \leq \varepsilon$$

for all $i \in u \setminus e_n^+$. Moreover:

$$d_i^{-\frac{1}{2}} \left\|\mathbf{f}_i^e - \mathbf{f}_i^*\right\|_\infty \leq \frac{1 - d_i^{-\frac{1}{2}} \left\|\mathbf{f}_i^e\right\|_\infty}{1 - \varepsilon}$$

for all $i \in e_n^+$.

Algorithm

In Section 3.2.2, we showed how to bound the difference between the harmonic solution on the graph W and its ε-subgraph $W[e]$. In this section, we propose an algorithm for building ε-subgraphs.

Algorithm 1 Incremental subgraph learning.

Input:
Graph W
Confidence level $\varepsilon \in [0, 1]$

$e^{(1)} \leftarrow l \cup \{i \in u \; \exists j \in l : w_{ij} > 0\}$
$t \leftarrow 1$
repeat
$L \leftarrow n \times n$ Laplacian of $W[e^{(t)}]$
infer $F_u \leftarrow -(L_{uu})^{-1} L_{ul} F_l$
$e^{+} \leftarrow \left\{i \in e^{(t)} : \|\mathbf{f}_i\|_\infty > \varepsilon\right\}$
$e^{(t+1)} \leftarrow e^{(t)} \cup \{i \in u : \exists j \in e^{+} \; (w_{ij} > 0)\}$
$t \leftarrow t + 1$
until $((|e^{(t)}| = |e^{(t-1)}|) \vee (|e^{(t)}| = n))$

Output:
ε-subgraph $W[e^{(t)}]$

The algorithm proceeds in iterations. First, the subgraph vertices $e^{(t)}$ are initialized to labeled vertices and their neighbors. Second, we compute the harmonic solution on the subgraph $W[e^{(t)}]$. Third, we find vertices e^{+} whose labels are inferred with sufficiently high confidence and add their neighbors to the subgraph vertices $e^{(t+1)}$. Finally, we compute the harmonic solution on the graph $W[e^{(t+1)}]$ and repeat all steps until the set $e^{(t)}$ stops growing. Our method is outlined in Algorithm 1.

Algorithm 1 can terminate for one of two reasons. First, $|e^{(t)}| = |e^{(t-1)}|$. In this case, all neighbors of highly confident predictions e^{+} are in $e^{(t)}$, and so $W[e^{(t)}]$ is an ε-subgraph. Second, $|e^{(t)}| = n$. In this case, $W[e^{(t)}]$ is the entire graph, which is an ε-subgraph for all ε. So Algorithm 1 always outputs an ε-subgraph. Note that the set $e^{(t)}$ increases monotonically, $e^{(t)} \subseteq e^{(t+1)}$. As a consequence, Algorithm 1 always terminates, for one of the above reasons.

The *confidence level* ε is the only parameter of Algorithm 1. The parameter ε controls the size of the resulting ε-subgraph $W[e]$. The smaller the value of ε, the larger the subgraph. We study this trend in Section 3.2.3.

Let $W[e^{(T)}]$ be the ε-subgraph generated by Algorithm 1. Then the space complexity of Algorithm 1 is $O(|e^{(T)}|^2)$, the space taken by $W[e^{(T)}]$. The time complexity is $O(T|e^{(T)}|^3 + n|e^{(T)}|)$, where T is the number of iterations, $O(|e^{(T)}|^3)$ is the cost of computing the HS in each iteration, and $n|e^{(T)}|$ is an upper bound on the total number of tests for neighbors of the

vertices e^+ when the set $e^{(t+1)}$ is generated. As a result, when $|e^{(T)}| \ll n$, the time complexity of our method is linear in n.

The above analysis makes no assumptions on $W[e^{(T)}]$. Suppose that $W[e^{(t)}]$ is $O(|e^{(t)}|)$ sparse for all t. Then the space and time complexity of our method are only $O(|e^{(T)}|)$ and $O(T|e^{(T)}|^2)$, respectively. Sparsity is common in practice.

Practical issues

The worst-case number of iterations in Algorithm 1 can be large. Consider the line graph example in Section 3.2.2 (Figure 3.4a). In this example, the labeled vertices are $n-1$ hops apart, and our method converges in $O(n)$ iterations irrespective of the confidence level ε, when the ε-subgraph covers the entire graph W.

Obviously, this behavior is undesirable. To avoid such cases, we suggest regularizing the harmonic solution as [69]:

$$F_u = -(L_{uu} + \gamma I_u)^{-1} L_{ul} F_l, \tag{3.16}$$

where I_u is a $n_u \times n_u$ identity matrix and γ is a tunable parameter. The regularizer γI_u can be viewed as a special *sink* vertex. At each step, the random walk on the graph W is absorbed in the sink with probability $\frac{\gamma}{d_i+\gamma}$. So the confidence $f_i[k]$ of labels decreases with the distance from labeled vertices l. In particular, note that:

$$f_i[k] \leq \left(1 - \frac{\gamma}{d_i + \gamma}\right)^{\tau_i}, \tag{3.17}$$

where τ_i is the number of hops between the vertex i and the closest labeled example. The above claim holds for any graph.

The parameter γ can be used to control the number of vertices in the subgraph. In particular, note that Algorithm 1 can add a vertex only if its neighbor is among the vertices e^+, which are defined as $e^+ = \left\{i \in e^{(t)} : \|\mathbf{f}_i\|_\infty > \varepsilon\right\}$. When:

$$\gamma = \left(\exp\left[-\frac{\log \varepsilon}{\kappa}\right] - 1\right) \max_i d_i, \tag{3.18}$$

$\|\mathbf{f}_i\|_\infty \leq \varepsilon$ for all vertices that are at least κ hops from the closest labeled vertex. This can be easily verified by substituting the above γ to Equation 3.17. As a result, Algorithm 1 never

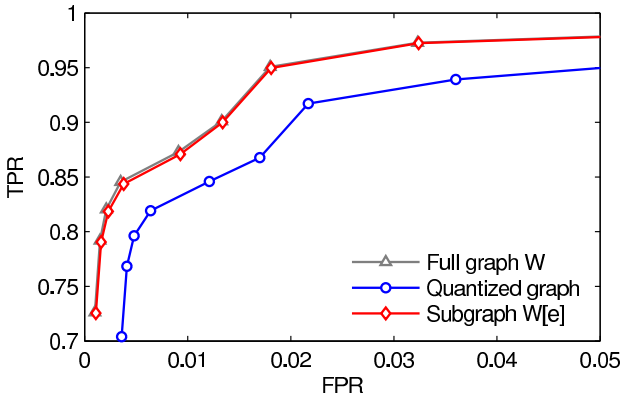

Figure 3.6: The TPR and FPR of three harmonic solutions on the digit recognition dataset.

adds a vertex that is more than k hops from the closest labeled vertex. In Section 3.2.3, we study how the performance of Algorithm 1 changes with γ.

Finally, note that Algorithm 1 requires the graph W to be sparse. Therefore, we consider similarity functions of the form:

$$w_{ij} = \begin{cases} \exp\left[-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right] & \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \phi \\ 0 & \text{otherwise,} \end{cases} \tag{3.19}$$

where the parameter ϕ controls the sparsity of W.

3.2.3 Digit Recognition

We now evaluate our approach on the digit recognition problem. The dataset and experimental setup are described in detail in Section 3.2.3. The digit recognition dataset is small and therefore we can build the entire data adjacency graph W. In addition, all data points are labeled. Therefore, we can easily evaluate the accuracy of our approach and compare it to baselines (Section 3.2.3). We also study the sensitivity of our method to the setting of its parameters (Section 3.2.3).

Experimental setup

The performance of our solution is evaluated on the problem of handwritten digit recognition [47]. The digit recognition dataset comprises of 5620 examples, digits between 0 and 9. Each digit is described by 64 features, which are discretized on an 8×8 grid.

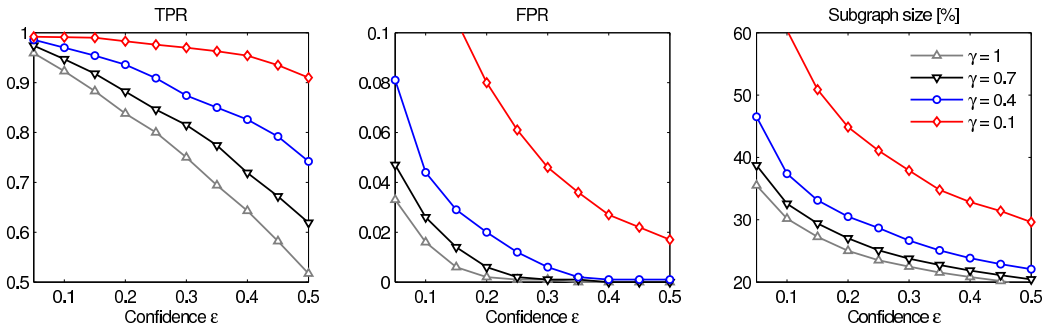

Figure 3.7: From left to right, we report the TPR and FPR of the HS on ε-subgraphs, and the size of the subgraphs $100\frac{|e|}{n}$, as functions of the confidence threshold ε. The subgraphs are built for four different values of γ.

The data adjacency graph W is constructed as described in Section 3.2.2. The heat parameter and sparsity threshold are set as $\sigma = 0.1\sqrt{K}$ and $\phi = 2\sigma$, where $K = 64$ is the number of features. In summary, the similarity of examples decreases exponentially with distance and is zero when the examples are more distant than 2σ. This results in a sparse graph W.

We cast digit recognition as a set of binary classification problems, one for each pair of the digits. In each problem, we label 10 randomly selected examples for each digit in the pair. The labels of other examples are inferred on the ε-subgraph $W[e]$. We regularize the Laplacian as described in Section 3.2.2 and set the regularization parameter γ to 0.1. Our results are averaged over all classification problems.

Comparison to baselines

The digit recognition dataset is small and therefore we can build the data adjacency graph W on this dataset. The accuracy of predictions on the ε-subgraph is expected to be suboptimal when compared to those on W. Perhaps surprisingly, we show that our predictions are nearly optimal. We also compare our solution to graph quantization (Section 3.2.1). In graph quantization, the graph W is approximated by a smaller graph on k representative examples. In our experiments, $k = |e|$ and therefore the quantized graph has the same size as the corresponding ε-subgraph. In other words, the HS on the quantized graph is of the same time and space complexity as the final iteration of Algorithm 1.

In Figure 3.6, we compare the TPR and FPR of the harmonic solutions on ε-subgraphs, the entire graph W, and the quantized graph. In ε-subgraphs, we only predict the labels of the

core vertices e^+. To make a fair comparison, the two baselines also predict only top $|e^+|$ most confident labels according to the HS. We vary the confidence level ε to get subgraphs of various sizes, and get a point on the ROC curve for each ε.

Figure 3.6 shows that our method makes almost as accurate predictions as the HS on the graph W. In fact, our results are so good that the corresponding ROC curves are hard to distinguish. Moreover, note that our method performs significantly better than graph quantization. This shows that smart allocation of vertices, by building ε-subgraphs, leads to better results than covering the graph by randomly chosen representative examples.

Sensitivity to parameter settings

In the second experiment, we study the sensitivity of the HS on the ε-subgraph to its two parameters, the confidence level ε (Section 3.2.2) and the regularization parameter γ (Section 3.2.2). We plot ROC curves for several values of γ and vary ε to get points on each curve. Our results are summarized in Figure 3.7.

In most cases, the TPR and FPR of the harmonic solution on ε-subgraphs are pretty high and relatively low, respectively. When ε is close to zero, the subgraphs cover a large portion of the graph and the TPR is at the maximum. As ε increases, the FPR and TPR decrease. Finally, when $\varepsilon = 0.5$, all subgraphs cover less than 30% of the entire graph and we make only confident predictions. Note that the minimal subgraph to predict correctly all instances of the two digits in each problem is about 20% of W.

The regularization parameter γ controls the size of ε-subgraphs (Section 3.2.2). As γ increases, fewer vertices are added to the subgraphs since the confidence of predictions decreases faster with the number of hops from labeled vertices. Therefore, the TPR and FPR decrease. Finally, note that ε-subgraph learning changes smoothly with γ and ε, and it is not sensitive to small perturbations of these parameters.

3.2.4 Topic Discovery

In this section, we tackle a large real-world problem of semantic inference or topic discovery over graphs generated from free-form text. We analyze two large datasets of restaurant

and hotel reviews, and infer multi-class membership probabilities over nouns and contextual descriptors in the corpora. It is computationally expensive to build a full graph over these datasets. Hence, our method is very useful in finding a relevant subgraph for the task at hand. In Section 3.2.4, we define the construction of the graph W that is approximated by our method. In Section 3.2.4, we introduce the domains of restaurant and hotel reviews, and describe our datasets. Our results are presented in Section 3.2.4 and we compare these with baseline methods in Section 3.2.4. Finally, we study the computational complexity of our algorithm in Section 3.2.4.

Data adjacency graph

We build a semantically coherent graph W over the text by utilizing the contextual descriptors around the words in the text. While semantically dissimilar words are often used in the same sentence, the descriptive context around the words creates a strong topical link. For instance, in our restaurant reviews the words "food" and "service" which belong to obviously different restaurant topics co-occur almost ten times as often as the words "food" and "chicken". However, we never expect to see the phrase "service is delicious", and we could use the contextual descriptor "___ is delicious" to link words under the food topic.

We build the subgraph over the textual data using Algorithm 1. Our graph comprises of two types of vertices - words and descriptors. The contextual descriptors consist of 1 to 5 words appearing before, after, or both before and after the words in review sentences. A five word window is sufficient to capture most commonly used phrases. There are millions of context descriptors and we consider only those which occur at least 20 times. In addition, we prune the list of descriptors to remove those with only stop words; a descriptor like "the ___" is not very informative. Secondly, our graph comprises of words that fit the descriptors. We restrict this step to finding nouns as in a sentence as the semantic meaning is often carried in the nouns. Therefore, we build a bipartite data adjacency graph W where the two parts correspond to words and their contextual descriptors.

The words and descriptors that co-occur in a sentence are linked by edges weighted by the point-wise mutual information (PMI) score [101, 26]. The edge weight between word i and

context descriptor j is:

$$w_{ij} = \log\left(\max\left\{\frac{P(i \cap j)}{P(i)P(j)} - \phi, 1\right\}\right), \tag{3.20}$$

where $P(i \cap j)$ is the probability that word i and context j appear together, $P(i)$ is the probability of the word i, and $P(j)$ is the probability of the context j. The value of $P(i)$ is estimated directly from data. Instead of computing $P(i \cap j)$ and $P(j)$, which would require normalization over all contexts, we estimate the ratio $\frac{P(i \cap j)}{P(j)}$ as the fraction of contexts j with the word i. The threshold ϕ controls the sparsity of the graph.

Datasets

We obtained two large user reviews datasets from popular online reviewing websites: the restaurant reviews dataset was mined from Yelp (`http://www.yelp.com`) and the hotel reviews dataset was mined from TripAdvisor (`http://www.tripadvisor.com`). Both these datasets have very different properties as described below and summarized in Table 3.2. Yet, our methods are easily applicable to these large and diverse datasets and our algorithm finds very precise semantic clusters as shown in Section 3.2.4.

The restaurant reviews corpus has 37k reviews with an average length of 9.2 sentences. The 344k sentences were used to compute the PMI for constructing the graph. The vocabulary in the restaurant reviews corpus is very diverse and contains several proper nouns like menu items and server names. We used the openNLP toolkit [3] for sentence delimiting and part-of-speech tagging to detect the nouns in the data. We ignore infrequently occurring misspellings and idiosyncratic word formulations, and retain the nouns that occur at least 10 times in the corpus. The restaurant reviews dataset contains 8482 distinct nouns. We defined five semantic categories over the text: food, price, service, ambience, social intent (describing the purpose of the visit). We used only a handful of labeled seed words for each class. The choice of the seed words was based on the frequencies of these words in the corpus as well as their generally applicable meaning to a broad set of words. The seed words used for inference are shown in the top portion of Table 3.3.

The hotel reviews dataset is much larger with 137k reviews on hotels in Europe, as shown in Table 3.2. Yet, the average number of sentences in a review is only 7.1 sentences and despite

	Restaurants	**Hotels**
Reviews	37224	137234
Sentences	344217	971739
Businesses	2122	3370
Users	18743	No unique user identifiers available
Distinct Nouns	8482	11212
Distinct Words	12080	19045

Table 3.2: Description of two large reviews datasets.

four times as many reviews as the restaurants corpus, the number of distinct nouns is only 11k. In the hotel reviews dataset, reviewers rate six different aspects of the hotel: Cleanliness, Spaciousness, Service, Location, Value and Sleep Quality. Assuming that these six semantic classes are important to users, we adhered to the same in our experiments. The labeled seed words used for each class are shown in Table 3.5.

ε-subgraph inference evaluation

We computed the HS on the ε-subgraph on the restaurant reviews dataset, and learned class labels for words and descriptors. The algorithm is parameterized as $\gamma = 1$, $\varepsilon = 0.3$, and $\phi = 128$. These parameters were chosen such that we explore only a small portion of the graph. Our algorithm quickly converges in 7 iterations and finds semantic confidence scores over 11% nouns. Table 3.3 shows the top 10 words with the highest class membership probability returned by the HS on the ε-subgraph. We observe that our method assigns high confidence scores to several synonyms of the seed words like *server, waitress, proprietor* for the service class and our method also captures common abbreviations and domain specific usage of words like *apps* and *starters* for the food class.

We do not have ground truth on the semantic meaning on words. Therefore, we manually evaluated the lists of top-K words with the highest confidence scores for belonging to each semantic group. We evaluated the performance of ε-subgraph inference using the precision@K metric for each semantic class. A high precision value indicates that a large number of the top-K words returned by the algorithm are labeled with the correct semantic class. Three judges

Labeled seed words				
food	**price**	**service**	**ambience**	**social intent**
food	price	service	ambience	boyfriend
dessert	cost	staff	ambiance	date
appetizer	costs	waiter	atmosphere	birthday
appetizers	value	waiters	decor	lunch
Top-10 discovered words by ε-subgraph inference				
food	**price**	**service**	**ambience**	**social intent**
starters	pricier	illusionist	downside	bday
apps	steamed	bruno	setting	graduation
howard	pricing	swan	general	lady
starter	tho	particular	presentation	husband
flatbreads	diego	server	sink	goodbye
error	source	proprietor	comstock	wife
don	crap	servers	vibe	farewell
sides	theres	waitress	impression	bachelorette
app	attitude	banter	uses	mom
desert	interpretation	tenders	octopus	sally

Table 3.3: Labeled seed words and top 10 words discovered by the HS on the ε-subgraph in the restaurant reviews domain.

evaluated the quality of the word classification (with inter-annotator kappa = 76%) and we used the majority vote for assessing the results. Table 3.4 shows the precision of the returned results for the five semantic classes at $K = \{5, 10, 20\}$. We see that at $K = 10$, we have a very high precision of over 90% for service and social intent and over 60% for food and ambiance. Our performance on the price class is poor because users rarely write about the price and rely on the metadata price classification of the restaurant.

Using the same parameter settings on the hotel reviews dataset, we run 5 iterations and explore a subgraph over 20% nouns and 14k descriptors, again only a small fraction of the complete graph over the text. Table 3.5 shows the top 10 words with the highest class membership probability returned by our ε-subgraph inference algorithm. Next, we evaluate the labels learned for the six semantic categories in our corpus. The precision@K for the ε-subgraph inference on the hotel reviews dataset is shown in Table 3.6. We have a high precision (above 70%) for all categories in this dataset with perfect precision for the service class. These results

Precision @	Semantic Class	ε-Subgraph Inference	Quantized Subgraph	Self Training
5	**food**	**0.8**	0.6	0.2
	price	0.4	0.4	**0.6**
	service	**0.8**	**0.8**	0.6
	ambience	**0.8**	**0.8**	0.6
	social intent	**1**	**1**	0.8
	Average	**0.76**	0.72	0.56
10	**food**	**0.7**	0.6	0.5
	price	0.2	0.4	**0.5**
	service	**0.9**	0.6	0.5
	ambience	0.6	0.6	**0.7**
	social intent	**1**	0.8	0.8
	Average	**0.68**	0.6	0.6
20	**food**	0.65	**0.75**	0.3
	price	**0.35**	0.3	**0.35**
	service	**0.9**	0.55	0.6
	ambience	0.55	0.55	**0.8**
	social intent	**1**	0.55	0.75
	Average	**0.69**	0.54	0.56

Table 3.4: Precision at top K semantic labels learned in the restaurant reviews domain.

are slightly better than the ones on the restaurant reviews dataset. We believe that the improvement in precision is due to the better defined and distinct classes in the hotels domain derived directly from TripAdvisor.

Comparison to baselines

We now compare the precision of our method with two baseline methods for semantic class labeling. First, we build a quantized subgraph (Section 3.2.1) by a random selection of vertices. We build the similarity graph using Equation 3.20, and compute the HS on this subgraph. In essence, inference on this quantized graph differs from our method only in the selection of the vertices to build the subgraph. For a fair comparison with our ε-subgraph inference method, we used the same number of noun and descriptor vertices as the subgraph found by our technique. Table 3.4 shows the precision@K for this quantized subgraph on the restaurant reviews dataset. As expected, we see an overall lower precision for the labels learned over the quantized subgraph in comparison with our method. At $K = 20$, while ε-subgraph inference generates high precision labels (> 0.9) for the service and social intent classes, quantization generates significantly lower precision (0.55).

As a second baseline, we adapt the self-training method from [82] called Espresso. As described in Section 3.2.1, the principle idea is that at each iteration Espresso finds new vertices in the data graph and deterministically assigns class labels to a few. The reliability metric described in [83] is used by Espresso to label vertices. This greedy method differs from our algorithm only in that it makes hard class decisions based on the reliability metric and there is no random walk inference computation. The construction of the similarity graph is identical to our implementation. As shown in Table 3.4, the Espresso self training algorithm provides less accurate class labels on the restaurant reviews dataset. Moreover, the computational complexity of such an algorithm is very high. Across all semantic classes, this self-training algorithm explored as many as 25% and 43% nouns in the restaurant and hotel reviews datasets only in the fourth iteration. Eventually, Espresso evaluated over 94% nouns. Hence, self-training methods achieve low precision and low efficiency in comparison to our ε-subgraph inference method. For all $K = 5, 10, 20$, the average precision across all five semantic classes is highest when using our ε-subgraph inference. At $K = 20$, we see a 28% and 23% improvement

Labeled seed words					
cleanliness	**service**	**spaciousness**	**location**	**value**	**sleep quality**
cleanliness	service	size	location	price	sleep
dirt	staff	closet	area	cost	bed
mould	receptionist	bathroom	place	amount	sheet
smell	personel	space	neighborhood	rate	noise
Top-10 discovered words by ε-subgraph inference					
cleanliness	**service**	**spaciousness**	**location**	**value**	**sleep quality**
accomodation	folks	sup	neighbourhood	package	noises
accommodation	clerks	wardrobe	someplace	airfare	pram
oder	attendants	vale	neighborood	deal	mattress
mold	workers	storage	schillerstrasse	tariff	sleeping
wonder	receptionists	fringe	recomend	sum	crash
odor	gals	warning	palce	alot	coins
mildew	personel	cupboard	hell	vs	jams
show	staffers	drawer	neigbourhood	lot	noice
hygiene	julie	counter	cul	charge	terror
stains	benedetta	shelf	intending	evaluation	pensionato

Table 3.5: Labeled seed words and top 10 words discovered by the HS on the ε-subgraph in the hotel reviews domain.

averaged across all classes by our method over quantization and self training respectively.

We now compare the inference of the alternate baseline methods on the hotel reviews dataset. The performance of the HS on the ε-subgraph is significantly better than the baselines of quantization and the self-training. Averaging across the six semantic classes at $K = 20$, we see a large improvement of 29% and 45% of our method over quantization and self training respectively. In the future, we wish to evalaute these alternate approaches with a much larger set of hand-labelled assessments.

Computational complexity

We now assess the gain in performance by using our method. We present results in the restaurant reviews domain; the hotels domain had similar gains. On the restaurant reviews dataset our algorithm runs for 7 iterations and finds semantic confidence scores over 11% nouns in the

Precision @	Semantic Class	ε-Subgraph Inference	Quantized Subgraph	Self Training
5	**cleanliness**	**0.8**	0.6	**0.8**
	service	**1**	**1**	0.8
	spaciousness	**0.8**	0.6	0.2
	location	0.8	**1**	0.6
	value	**1**	0.8	0.6
	sleep quality	**1**	0	0.6
	Average	**0.9**	0.8	0.6
10	**cleanliness**	**0.8**	0.7	0.7
	service	**1**	**1**	0.7
	spaciousness	**0.8**	**0.8**	0.5
	location	**0.8**	**0.8**	0.5
	value	**0.9**	0.6	0.6
	sleep quality	0.7	**0.9**	0.4
	Average	**0.83**	0.8	0.57
20	**cleanliness**	**0.8**	0.65	0.7
	service	**1**	0.75	0.75
	spaciousness	**0.7**	0.65	0.5
	location	**0.8**	0.6	0.65
	value	**0.85**	0.55	0.5
	sleep quality	**0.7**	0.6	0.25
	Average	**0.81**	0.63	0.56

Table 3.6: Precision at top K semantic labels learned in the hotel reviews domain.

corpus. To evaluate the reduction in computational cost, we generated all context descriptors in the corpus using the neighborhood window around all occurrences of words and found that our corpus contains 41k frequent descriptors (occurring at least 20 times). In comparison, our algorithm generates a subgraph comprising less than 5k descriptors. Thus, we explore only a small fraction of the complete graph that is most semantically meaningful.

The computation time of the topic discovery experiments is dominated by queries to the database that store the sentences from the reviews. At every iteration, we expand highly confident vertices e^+. This involves retrieving sentences from the database containing the vertices and finding new neighboring vertices in the text. The set e^+ comprises of only 0.7% nodes in the restaurant reviews domain, resulting in significant improvement in computation time. In addition, the ε-subgraph $W[e^{(T)}]$ comprises of only 11% nodes and the space required $O(|e^{(T)}|^2)$

is two orders of magnitude smaller than the full graph built on all vertices (Section 3.2.2). However, our algorithm requires finding the HS for each iteration. Yet, our experiments require very few iterations in general, 7 and 5 iterations over the restaurant and hotel domains respectively. Therefore, the dominating time is the database queries for finding neighbors of vertices and generating the adjacency matrix. Overall our algorithm achieves significant performance gain over a one-shot inference on the full graph.

3.3 Conclusions

In this chapter, we presented the user reviews supervised classification and analysis effort performed as part of our URSA (User Review Structure Analysis) project. We show that both topic and sentiment information at the sentence level are useful information to leverage in a review. We developed techniques for manual annotation of labeled data and automatic sentence classification over the review sentences with both the topic and sentiment classes. Our SVM-based classifiers yield highly accurate results, augmenting free-form text with structure useful for automatic processing.

Additionally, we described a highly efficient semi-supervised algorithm for topic discovery that does not require large amount of human input. The harmonic solution on a graph is a popular approach to semi-supervised learning. Unfortunately, the method does not scale well with the size of training data n because its space and time complexity are $\theta(n^2)$ and $\theta(n^3)$, respectively. In this chapter, we studied a new approach to approximating the harmonic solution and we show how highly confident HS predictions on a graph can be identified based on a subgraph. We demonstrated the performance of our method in obtaining nearly optimal semantic labels over words in a graph over user reviews in the restaurant and hotel reviews domain.

In the following chapter, we leverage the topical and sentiment information from user reviews to build a text-based recommendation system. Our experiments show that textual information, as captured by the classification techniques described above, allow making for highly accurate and fine-grained predictions of user preferences.

Chapter 4

Text-based Recommendations

Today, web users have wholeheartedly incorporated peer-authored product reviews into their daily decisions. Yet, web sites providing user reviews are surprisingly technologically poor: users often have no choice but to browse through massive amounts of text to find a particular piece of relevant information.

Accessing and searching text reviews is particularly frustrating when users only have a vague idea of the product or its features and they need a recommendation or closest match. Keyword searches typically do not provide good results, as the same keywords routinely appear in good and in bad reviews [10]. Another challenge in understanding reviews is that a reviewer's overall rating might be largely reflective of product features in which the search user is not interested, as demonstrated in Example 1.

Ideally, users should not have to read through several reviews, but should be presented with items that they would find interesting or useful based on some notion of preference through similarity with other users or items. This task of preference matching is carried out by recommendation systems [21]. Current recommendation systems such as the ones used by Netflix or Amazon [74] rely predominantly on structured metadata information to make recommendations, often using only the star ratings, and ignore a very important information source available in reviews: the textual content.

We propose techniques that harness the rich information present in the body of the reviews by identifying the review parts pertaining to different product features (e.g., food, ambience, price, service for a restaurant), as well as the sentiment of the reviewer towards each feature (e.g., positive, negative or neutral) and leverage this information to improve user experience.

Identifying such structured information from free-form text is a challenging task as users routinely enter informal text with poor spelling and grammar. In the previous chapter, We performed an in-depth classification of a real-world restaurant review data set using supervised classification with topical and sentiment classes. Our work addresses categorization and sentiment analysis at the sentence level as web reviews are short and designed to convey detailed information in a few sentences. In this chapter, we apply our text classification to a recommendation scenario and show that the rich textual information can improve rating prediction quality. In addition, we propose methods to predict the sentiments of users towards individual restaurant features and enhance user experience by presenting the review parts pertaining to these features.

Our work, performed as part of the **URSA** (User Review Structure Analysis) project, takes the novel approach of combining natural language processing, machine learning and collaborative filtering to harness the wealth of detailed information available in web reviews [43]. Our techniques utilize the free form textual data from user reviews for collaborative filtering, a domain where most studies have focused on using ratings and other structured metadata. In this chapter we present personalized recommendation techniques that use the full text of a review to make ratings predictions as well as predictions on user sentiment towards restaurant features. In particular we make the following novel contributions:

- We implement a new quadratic regression model using all of the detailed textual information obtained from the text classification, to derive text-based ratings (Section 4.2.2). In comparison with the simple ad-hoc and linear regression presented in [42], the quadratic model is a better fit for our data (estimated by the lowered error in regression), and yields more accurate rating predictions.

- We compare the predictive power of star and textual ratings using average-based strategies that incorporate the rating behavior of the user, the average quality of the restaurant, and a combination of both (Section 4.2.3).

- We reviewed state of the art recommendation techniques and evaluated their performance on our restaurant reviews corpus. As described in Section 4.3.1, rating-based methods using latent factorization and neighborhood models do not yield significant improvements

over average-based baseline predictions for our sparse dataset.

- We utilize the rich textual information present in the reviews to better group similar users for making recommendations. Users who have reviewed common restaurants are clustered together if they have liked or disliked the same aspects of the restaurant in the text of their reviews, thus providing an approach to address the problem outlined in Example 1. We implement a text-based soft clustering of users and design a novel prediction approach for making personalized predictions in Section 4.3.2.

- We present an approach to predicting not just a numeric rating, but the sentiments of users towards individual restaurant features (Section 4.4).

We described our restaurant reviews data set in Chapter 3 and discussed our text classification approach. We utilize the supervised classification model identifying six topics and four sentiment classes over the sentences in restaurant reviews. We now utilize this structure augmented text to build a text-based recommendation system over user authored reviews.

The techniques in this chapter have been published in [43]. This chapter is structured as follows. We describe the evaluation settings for our prediction experiments in Section 4.1. In Section 4.2, we propose new regression-based measures that take into account the textual component of reviews for deriving alternate text-based ratings for user reviews. We then turn our focus to accurately predicting ratings for making useful recommendations, using average-based recommendation strategies. In Section 4.3, we evaluate popular rating-based methods like matrix factorization and KNN and evaluate the use of the textual information for clustering like-minded users in personalized prediction settings. We show that relying on user reviewing behaviors, as determined by the type of sentences covered in the reviews, results in an improvement in predictions over techniques that only consider ratings. We then use the textual information to predict user sentiments towards individual restaurant features in Section 4.4. We conclude in Section 4.5.

4.1 Recommendation System Evaluation Setting

To evaluate the predictive value of our recommendation methods, we randomly extracted three test sets of around 260 reviews each from the restaurant data set; the remaining reviews comprised the corresponding training sets. A review set aside in the test set is not used in making predictions, but is only used in evaluating the accuracy of the predictions. For personalized recommendations, we are interested in using user-specific information for clustering users, and we need at least one review written by the users in the test set to derive user-specific information. Therefore, two of our test sets – A and B, are randomly chosen such that each test user has at least one review in the training set in addition to the review set aside for the test set.

Test set C contains one review each from users who have rated at least 5 restaurants. Therefore, Test set C contains more usable user-specific information than the randomly chosen Test sets A and B.

Our data contains only the review date information and no time stamp. A majority (86%) of users have written all their reviews on the same day. Hence, we are unable to create test sets containing the last review written by the user, as is often done, e.g. the Netflix Challenge test set [19].

We now focus on predicting ratings for the test set reviews using baseline average-based strategies.

4.2 Predicting Restaurant Ratings

In this section, we first describe our methodology (Section 4.2.1) for predicting the overall ratings for the reviews in the test sets. To compare the use of star ratings with the textual data in a recommendation scenario, we propose a novel method for deriving textual ratings using our sentence classification in Section 4.2.2. Textually derived ratings serve as an alternate assessment in the review based on the user sentiment towards different product aspects. We then evaluate the predictive utility of the star ratings and the text-based ratings using average-based prediction strategies in Section 4.2.3. Note that in Section 4.4, we go beyond the goal of predicting the overall review rating and focus on making fine-grained predictions on user sentiment towards individual restaurant aspects.

4.2.1 Methodology

Our goal is to use the information present in the training data to accurately predict the ratings in the test set. To explore whether the text in reviews is a better predictor of user assessment of a restaurant than the star ratings, we derive an alternate textual rating from the body of the reviews as described in Section 4.2.2. Using this analysis, we have two alternate methods to manipulate the information present in the training data: the star ratings in the reviews, and the textual ratings derived from the body of the reviews.

In addition, the reviews in the test set also contain both star ratings and textual ratings. Therefore, we have two prediction goals: accurately predicting the star ratings of the test set reviews and accurately predicting their text ratings.

We use the popular root mean square error (RMSE) accuracy metric to evaluate our prediction techniques [52].

4.2.2 Textually Derived Ratings

The text of a review (as approximated by its associated topics and sentiments) can enable us to capture the detailed assessment by a user of the restaurant. We use a regression-based method for deriving textual ratings from the review text as described below.

Regression-based Method

Typically, users assign different degrees of importance to the topics of their reviews. For each review in the corpus, we propose a textual rating which incorporates topics and sentiments with varying levels of importance into a regression-based rating. Regression allows us to learn weights to be associated with each sentence type. These weights are learned from the data set itself, and therefore closely represent how people write reviews in a domain. Our regression models the user-provided star ratings as the dependent variable; the sentence types represented as *(topic,sentiment)* pairs are the independent variables, i.e., we performed a multivariate regression which learns weights or importance to be associated with the different textual information (represented by the several sentence type variables).

We computed the multivariate regression using the least squares estimates method. We

Constant	3.68			
1^{st} **Order Variables**	Positive	Negative	Neutral	Conflict
Food	2.62	-2.65	-0.078	-0.690
Price	0.395	-2.12	-1.27	0.929
Service	0.853	-4.25	-1.83	0.358
Ambience	0.747	-0.269	0.162	0.215
Anecdotes	0.957	-1.75	0.061	-0.186
Miscellaneous	1.30	-2.62	-0.303	0.358
2^{nd} **Order Variables**	Positive	Negative	Neutral	Conflict
Food	-2.00	2.04	-0.134	0.664
Price	-0.265	2.03	2.26	-1.01
Service	-0.526	3.15	1.79	0.354
Ambience	-0.438	0.801	-0.263	-0.595
Anecdotes	-0.401	1.97	-0.081	-0.262
Miscellaneous	-0.651	2.38	0.492	-0.089

Table 4.1: Four-sentiment regression weights.

performed a qualitative comparison between different sentence types settings and varying regression models, as described in the following section.

Four-Sentiment Second-Order Regression

A important step when fitting data is to find a good regression model. We experimented with linear multivariate models, as well as second order and third order models. The goodness of fit for these models is estimated using the root mean squared error for the regression [78].

We observed that the quadratic regression model, incorporating all the textual features (six topics and four sentiments), is a better fit for the restaurant reviews data set than the earlier proposed linear model in [42] (as estimated by the lowered error in regression). Our quadratic regression model for deriving textual ratings has the general form for three independent variables shown in Equation 1.

$$y \pm \phi = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_1^2 + \beta_5 x_2^2 + \beta_6 x_3^2 \tag{4.1}$$

In the above, $\beta_0, \beta_1, \ldots, \beta_6$ are the unknown weights that we wish to determine. x_1, x_2, x_3 are the sentence types frequencies. The dependent variable y is the star rating. ϕ is the error in regression, a good model will have a low error.

		Restaurant Average			User Average			Combined		
		TestA	**TestB**	**TestC**	TestA	TestB	TestC	TestA	TestB	TestC
Predicting Star rating	Using Star rating	**1.127**	**1.267**	**1.126**	1.313	1.349	1.061	1.283	1.363	1.095
	Using Text rating	**1.126**	**1.224**	**1.046**	1.149	1.231	1.035	1.143	1.236	1.029
Predicting Text rating	Using Star rating	**0.703**	**0.718**	**0.758**	0.971	0.969	0.649	0.990	1.031	0.812
	Using Text rating	**0.545**	**0.557**	**0.514**	0.603	0.631	0.491	0.609	0.637	0.523

Table 4.2: Prediction RMSE using average-based methods.

This model uses all information derived from the text classification which is beneficial for building a robust system. We build our model on the 50K examples in the training set as described in Section 6.3.1. Note that our model provides a regression constant which serves as a default rating when no textual information is available. The constant of 3.68 is slightly skewed towards a good review (star rating 4 or 5); this is consistent with the distribution of star ratings in the restaurant review corpus as discussed in Chapter 3. Finally, the second-order weights (shown in Table 4.1) have the reverse polarity as the corresponding first-order weights: the second order variables tend to dampen the effects of the first-order variables if many sentences of a type are present in a review.

The weights for our quadratic regression model are shown in Table 4.1. The proportion of Positive and Negative sentiment sentences have a clear effect on the rating in a review, as shown by the highly polar regression weights for these sentiments. As expected, the weights confirm that the Food category has the highest impact on the perception of a restaurant. The weights of the negative Price and Service related sentences are quite significant, indicating that unacceptable prices or poor service in a restaurant has a very adverse impact on the dining experience of users.

4.2.3 Average-based Predictions

We now focus on making predictions for the reviews in our test sets, using three average-based techniques. Our methods use the average assessment of the restaurant, the average rating behavior of the user, and a combination of both. For each strategy, predictions using text ratings provide better predicting accuracy (lower RMSE values) as compared to the predictions using the star ratings as shown in Table 4.2.

In the restaurant average-based prediction technique the rating of a test review is predicted

as the average rating of all the other reviews for the test restaurant. The resulting RMSE values are shown in the leftmost columns of Table 4.2. For the task of predicting star ratings, there is a significant improvement in prediction accuracy (7.1% and 3.4%) achieved for Test sets B and C, when textual ratings are used for making predictions.

For predicting textual ratings, the text again always outdoes the star ratings in making accurate predictions. Textual ratings indicate the general preference of a user towards the restaurant. However, information in the review about the sentence topics and sentiments are combined in a single rating. Predicting text ratings coarsely predicts the textual component of a review but does not predict individual topics and sentiments that are likely to be in the review. We will focus on such detailed qualitative predictions in Section 4.4. Note that the textual ratings have a lower standard deviation and therefore average-based strategies for predicting text ratings are expected to have lower errors.

We next examine the user average-based prediction strategy where the predicted value is the average rating of all the other reviews written by the test user (second column of Table 4.2). Lastly, we use a combination method where the predicted rating uses the deviation of the user average and the restaurant average from the data set average rating, as suggested in [66]. The results for this combined average-based method are included in the rightmost columns of Table 4.2. For Test set C, where users have reviewed many restaurants, user average or combined average prediction strategies prove to be less erroneous than the aforementioned restaurant average strategy. However, a large majority of users do not write many reviews (78% users have written only one review). The restaurant average predictions performs better in the generalized setting. Thus, we use the restaurant average approach as our baseline.

The results in Table 4.2 show that for each of the three average-based prediction strategies, using our textual ratings has a considerable advantage for making accurate predictions over the star ratings. We now focus on making better predictions using personalized recommendation strategies by finding like-minded users.

4.3 Personalized Rating Prediction

A limitation of the prediction metrics presented in the previous section is that they do not take advantage of all the usable information: the restaurant average prediction strategy results in all users receiving the same prediction for a restaurant regardless of individual preferences. In order to make better and personalized predictions there is a need to leverage information beyond the restaurant average by taking into account the similarities between users.

In this section, we investigate personalized recommendation techniques. In Section 4.3.1, we implement two popular state of the art collaborative filtering methods that rely on ratings (either star ratings or textually derived scores) for making predictions. In Section 4.3.2 we demonstrate the utility of our sentence classification for making accurate recommendations. We not only use textually derived ratings, but also utilize the textual patterns in user reviews for a grouping or clustering of similar users using a text-based soft clustering of users.

4.3.1 Rating-based Personalized Prediction

In recent years, there have been several studies on collaborative filtering models that rely predominantly on the ratings given by the users to the different items to make predictions. Such models saw a surge in popularity during the Netflix challenge [19]. In this section, we implement two ratings-based methods for making personalized predictions. In Section 4.3.1, we first implement a factorization method on the matrix of ratings in the training set to uncover latent features for predicting the ratings in the test sets. Next, in Section 4.3.1 we implement a neighborhood-based model for grouping or clustering of similar users.

Latent Factor Model

Matrix factorization (MF) has been useful for collaborative filtering in several previous studies [17, 16, 97, 104, 67] due to its ability to discover latent factors underlying the ratings given by the users to the items. These latent factors can then be used to predict unknown ratings. For a $m \times n$ matrix R comprising ratings given by m users to n restaurants, MF approximates R with the best rank-k approximation $\hat{R_k}$. $\hat{R_k}$ is computed as the product of two matrices $P_{m\times k}$ and $Q_{n\times k}$. In other words, to approximate R we factorize it into two low dimensional matrices

P and Q (typically $k << min(m,n)$) such that $\hat{R_k} = PQ^T$ or $R \approx PQ^T$.

MF associates each user i with a user-factors vector P_i of size k representing the underlying latent factors explaining user ratings, similarly each restaurant j with a vector Q_j. To find the suitable factors P and Q we apply a gradient descent method [97, 67]. We start with randomly initializing $P_{m\times k}$ and $Q_{n\times k}$, and calculate how different their product $\hat{R_k}$ is from R for the known ratings. Note that R is a very sparse matrix, with zeros representing missing or unknown ratings. Let (i,j) represent the ℓ known ratings in the dataset given by users i to restaurants j. The basic form of the squared approximation error is computed as follows:

$$\begin{aligned} {e_{ij}}^2 &= (R_{ij} - P_i {Q_j}^T)^2 \qquad for (i,j) \in \ell \\ &= \sum_{(i,j)\in\ell} \left(r_{ij} - \sum_k p_{ik} q_{kj} \right)^2 \end{aligned} \tag{4.2}$$

To avoid over fitting, we apply regularization to the basic form [17, 97] by penalizing with the magnitude of the user vector P_i and restaurant vectors Q_j. We introduce the regularization parameter λ which controls the magnitude of the vectors P_i and Q_j such that they would be a good approximation of R without containing large numbers. Therefore, the error is computed as:

$$e'_{ij} = \frac{1}{2} \left({e_{ij}}^2 + \lambda (\|P_i\|^2 + \|Q_j\|^2) \right) \tag{4.3}$$

We iteratively reduce the error in Equation 4.3 by implementing a gradient descent method to find a local minimum on the error. We compute the gradient of e'_{ij} for each k as follows:

$$\frac{\partial}{\partial p_{ik}} e'_{ij} = -e_{ij}.q_{kj} + \lambda.p_{ik}, \frac{\partial}{\partial q_{kj}} e'_{ij} = -e_{ij}.p_{ik} + \lambda.q_{kj} \tag{4.4}$$

Therefore, in each iteration we change the values in P and Q to decrease the approximation error. The change in the values is in small steps controlled by α, as follows:

$$\begin{aligned} p'_{ik} &= p_{ik} + \alpha.(e_{ij}.q_{kj} - \lambda.p_{ik}) \\ q'_{kj} &= q_{kj} + \alpha.(e_{ij}.p_{ik} - \lambda.q_{kj}) \end{aligned} \tag{4.5}$$

		TestA	TestB	TestC
Predicting Star Rating	Using Star rating	1.187	1.270	1.146
	Using Textual rating	1.148	1.215	1.083
Predicting Textual rating	Using Star rating	0.856	0.913	0.838
	Using Textual rating	0.630	0.640	0.599

Table 4.3: Prediction RMSE using matrix factorization for personalized predictions based on ratings.

We implemented the above mentioned regularized MF with gradient descent on our restaurant reviews dataset. For our dataset a rank 20 approximation with the regularization parameter λ set to 0.2 gave us the lowest RMSE errors. Table 4.3 shows the errors in predicting the ratings on the three test sets. We observe that matrix factorization does not yield better results than our restaurant average-based strategy (Section 4.2.3). Our dataset is very sparse and a large number of rows and columns in the ratings matrix have almost all zero entries. Previous studies [17, 16, 97, 67] showed the usefulness of MF on the Netflix Challenge data [19], which has 40 times more known ratings in the training set as compared to our corpus. From the results in Table 4.3, we see that MF does not perform well in very sparse scenarios. Note that matrix factorization captures both user and restaurant biases, and should more fairly be compared with the combined averages method of Section 4.2.3. In comparison to this baseline strategy, for the general Test Sets A and B the personalized prediction using MF performs marginally better.

Latent factor models have been successfully used in several previous studies [17, 16, 97, 104, 67]. However, MF does not yield sufficiently low errors on our sparse restaurant reviews corpus. In addition, latent factor models have low explainability; the meaning of the discovered latent factors is unclear. In the following section we experiment with neighborhood-based methods by grouping users based on the similarities in their rating behaviors.

Neighborhood Model

Our dataset has many more reviews for each restaurant on average than the average number of reviews per user. As a result, a restaurant average-based strategy performs well on our corpus as shown in Section 4.2.3. Therefore, we now focus on grouping similar users and make the prediction as the weighted average of the ratings given to the test restaurant by close neighbors.

		TestA	TestB	TestC
Predicting Star Rating	Using Star rating	1.130	1.259	1.124
	Using Textual rating	1.125	1.224	1.048
Predicting Textual rating	Using Star rating	0.704	0.719	0.767
	Using Textual rating	0.543	0.559	0.514

Table 4.4: Prediction RMSE using KNN for personalized predictions based on ratings.

We consider a K-Nearest Neighbor algorithm (KNN), a popular collaborative filtering technique [52], to identify the closest neighbors to a test user. After empirically comparing several distance functions, we computed the neighbors using a Pearson distance function with threshold [89] (our implementation uses a threshold value of 5). The threshold accounts for the number of items in common between users so that users are not considered as very close neighbors on the basis of only one common restaurant rated similarly.

The prediction algorithm uses the average of the K closest neighbors' scores (star rating or text rating) for the target restaurant as the predicted score. If a neighbor has not reviewed the restaurant, it uses the restaurant average-case prediction (Section 4.2.3) for that user.

We experimentally observed that the closest predictions were made when a close neighborhood of three users was used ($k = 3$). The resulting RMSE values are given in Table 4.4. The results are comparable to the baseline restaurant average-based prediction of Section 4.2.3; using close neighbors based on star or textual rating information does not help in improving rating predictions. In our sparse data users tend to review few restaurants making it difficult to find good neighbors that have reviewed the same restaurants and given similar ratings (cold start problem).

Using only the coarse ratings (star ratings or textually-derived ratings) for clustering is very restrictive. While the text-based ratings are derived using our sentence classification, all the information is combined into a single rating, making it difficult to distinguish the individual topics and sentiments covered in the review. Therefore, there is a need to use the full detailed textual information for finding like-minded users to make better personalized predictions, as described in the next section.

4.3.2 Text-based Personalized Prediction

We now explore enhanced techniques for finding similar users via clustering that utilize the textual information gained from the topical and sentiment classification. Unlike a hard clustering of users that assigns each user to exactly one cluster, soft clustering techniques assign users to every cluster with a probability greater than or equal to 0, and the sum of cluster membership probabilities for a given user equals to 1. There is evidence in the literature that in comparison to hard clustering, soft clustering is more robust to noise and performs better when the data cannot be separated into distinct clusters [40, 73]. Textual data is often fuzzy and a recommendation system built on such data will benefit from using probabilistic techniques for smoothening misclassification errors. Soft clustering captures the uncertainty in assigning values to clusters due to the similarity of values [31]. It also allows users to belong to different clusters with various degrees of confidence, allowing to represent for instance, user taste for both fine French cuisine and cheap Chinese dim sum. Therefore, we choose to implement a soft-clustering of the users to find similar users.

We use the Information Bottleneck (IB) Method [92] that assigns a probability to each user to belong to every cluster. The IB principle is described briefly in Section 4.3.2. In Section 4.3.2, we describe our adaptation of the iterative information bottleneck (iIB) algorithm [92] for clustering. We describe our novel prediction strategy using the cluster membership probabilities of users gained from the iIB algorithm in Section 4.3.2. The effects of parameter selections for the iIB method on the accuracy of ratings predictions are described in Section 4.3.2. Finally, after laying the groundwork, we describe experiments using the textual information in reviews as features for clustering in Section 4.3.2 and compare the prediction accuracy with the baseline restaurant average strategy of Section 4.2.3.

Information Theoretic Clustering

The Information Bottleneck (IB) method was first introduced in [99] as an information-theoretic approach for data analysis and clustering. This method has been successfully used in document classification [93], unsupervised image clustering [48] and many other applications. We use the

	R_1	R_2	R_3
U_1	4	-	-
U_2	2	5	4
U_3	4	*	3
U_4	5	2	-
U_5	-	-	1

Table 4.5: Example: Matrix of ratings given by five users to three restaurants.

	R_1				R_2				R_3			
	Food Pos	Food Neg	Price Pos	Price Neg	Food Pos	Food Neg	Price Pos	Price Neg	Food Pos	Food Neg	Price Pos	Price Neg
U_1	0.6	0.2	0.2	-	-	-	-	-	-	-	-	-
U_2	0.3	0.6	0.1	-	0.9	-	0.1	-	0.6	0.1	0.2	0.1
U_3	0.7	0.1	0.15	0.05	-	-	-	-	0.2	0.8	-	-
U_4	0.9	0.05	0.05	-	0.3	0.4	0.2	0.1	-	-	-	-
U_5	-	-	-	-	-	-	-	-	-	0.7	0.3	-

Table 4.6: Example: Matrix with four features as input to iIB algorithm.

IB method in a collaborative filtering scenario to find similarity between users. The main principle behind the Information Bottleneck clustering is that the data is clustered or compressed such that the new compressed representation of the data retains the maximum possible amount of relevant information present in the data.

Let X be a discrete random variable distributed according to $p(x)$; the variable X represents the objects to be clustered. X contains information about another variable: the relevant variable Y. The goal of any clustering method is to cluster the data points in X such that the resulting clusters maintain most relevant information about Y. Let T, another random variable, denote the compressed or clustered representation of X. A soft clustering, as achieved using the IB method, is defined though a probabilistic mapping of each value $x \in X$ to each value $t \in T$. Therefore, the final output of the IB method is the membership probabilities of the data points in X in each of the clusters T.

The IB principle has its roots in Rate Distortion Theory. There can be several possible clusterings of the input variable X into the new representation T. One goal of clustering is to compress X, or to represent the input data points using a small number of clusters. Thus,

	c_1	c_2	c_3
U_1	0.04	0.057	0.903
U_2	0.396	0.202	0.402
U_3	0.38	0.502	0.118
U_4	0.576	0.015	0.409
U_5	0.006	0.99	0.004

Table 4.7: Example: Cluster membership probabilities generated by the iIB algorithm.

the quality of the new representation T can be measured by its compactness. However, the compression is not enough. The *compression measure* can always be improved by ignoring details in X (e.g., by grouping all users in a single cluster), which will imply that the new representation T loses all relevant information about Y. Therefore, an additional constraint is needed; a *distortion measure* which represents the distance between the random variable X and its new representation T. The trade-off between the compactness of the new representation and its expected distortion is the fundamental trade-off in rate distortion theory.

Using the compression-distortion trade-off, the IB method aims to minimize the mutual information between X and its compressed representation T *(compression measure)*, under some constraint on the minimum mutual information that T preserves about the relevant variable Y *(distortion measure)*. In this sense, one is trying to squeeze the information X provides about Y through the compact "bottleneck" formed by the compressed representation T [92].

The trade off between the compression and distortion is parameterized by a single Lagrange parameter β. A large value of β ($\beta \to \infty$) indicates that the focus of the clustering is on the relevance of the underlying data, and the compression achieved through clustering is immaterial. In this case, each data point is put in a separate cluster of its own. On the contrary, a small value of β ($\beta \to 0$) assigns all data points in the same cluster, achieving maximum compression.

A detailed explanation of the IB method and the various algorithmic implementations can be found in [92]. In particular, we adapted the iterative information bottleneck (iIB) algorithm, and describe our implementation for the restaurant reviews data set in the following section.

Iterative Optimization Algorithm

We used the Iterative Information Bottleneck (iIB) algorithm, introduced in [99], to cluster like-minded users based on their reviewing behavior. As mentioned earlier, the goal is to cluster the input variable X via a probabilistic mapping to the variable T; while ensuring that T maintains maximum possible information about Y. Thus, in our case, the variable X represents the 30K users in our corpus. We use the different sentence types obtained from the text classification, represented as *(topic, sentiment)* pairs, as features for clustering. Therefore, the relevant variable Y represents the user preferences modeled by the information conveyed in the text of the reviews.

Consider the following artificial example consisting of a corpus of five users and three restaurants. The matrix representing the ratings given by these users to the restaurants is shown in Table 4.5; a blank matrix entry m_{ij} indicates that the corpus contains no review by user U_i for the restaurant R_j. Also, the $*$ in the m_{32} cell of the matrix indicates that we wish to predict the rating given by U_3 for R_2.

For simplicity, suppose that we cluster the five users based on only four sentence types; positive and negative sentences belonging to the food and the price categories (in actual experiments, all combinations of the six topics and four sentiment classes are used). This textually derived information is represented in a matrix shown in Table 4.6. The matrix shows that in the review written by U_1 for R_1 with 5 sentences, 3 were positive sentences about food, 1 was a food-related negative sentence and there was 1 positive price-related sentence. The entries in the matrix for each of the features is the normalized number of sentences of the feature type. (In actual experiments, the input matrix $P(X, Y)$ is a joint probability matrix, which is obtained from the matrix of restaurant-wise sentences of each type written by the users, similar to Table 4.6, after first ensuring that the sum of all entries in each row is 1, and then normalizing such that the sum of all entries in the matrix is 1.)

Given the input joint probabilities $P(X, Y)$, the iIB algorithm starts with a random initialization of the cluster membership probabilities $p(t|x)$. It then iteratively updates the probability matrix and converges to stable probability estimates [92]. The resulting output of the algorithm at the end of n iterations is a matrix $p^n(t|x)$ containing the *membership probabilities* of each

user for each cluster.

Now, suppose we wish to cluster the users in Table 4.5 into 3 soft clusters. For our example the output matrix is shown in Table 4.7. As expected U_2 and U_3 are somewhat similarly clustered, while the clustering of U_1 or U_5 is distinct from all other users. These membership probabilities are then used for making personalized rating predictions (Section 4.3.2).

We experimented with several values for the cluster cardinality M and the trade-off parameter β. We use a sufficiently large value for the cluster cardinality ($M = 300$) and set $\beta = 20$. A brief comparison of the effects of parameter selection on prediction accuracy is outlined in Section 4.3.2.

Personalized Prediction Strategy

We now describe our novel rating prediction strategy based on a soft clustering of users. The output of the iIB algorithm is a soft clustering of the users X into T clusters with the probabilities given in $P^{(n)}(t|x)$, similar to Table 4.7. We use these probabilities to find the weights to be associated with the users who have reviewed the restaurant of interest, i.e., the restaurant in the test case. The predicted rating for a test case is the weighted average of the ratings of all other users who have reviewed the restaurant.

The weights model the similarities between users. Users who have similar cluster membership probabilities across all clusters are close neighbors. For each cluster, we first compute the cluster contribution as the weighted average of the ratings of all users who have reviewed the test restaurant. Formally, suppose we want to predict the rating given by the test user U_t to the test restaurant R_t. Let $Pr(U_t, R_t)$ denote this prediction. Assume that n users have reviewed the test restaurant with ratings: $rating(U_1, R_t)$, $rating(U_2, R_t)$, ..., $rating(U_n, R_t)$. Also, for each user, U_1, U_2, ..., U_n who has reviewed the test restaurant, let $U_1(c_i)$, $U_2(c_i)$, ..., $U_n(c_i)$ denote the probabilities with which these users belong to a cluster c_i. Now, the contribution for a cluster c_i is given by:

$$Contribution(c_i, R_t) = \frac{\sum_{j=1}^{n} U_j(c_i) * rating(U_j, R_t)}{\sum_{j=1}^{n} U_j(c_i)} \tag{4.6}$$

Furthermore, we have M clusters, say c_1, c_2, ..., c_m. The final prediction for the test

review takes into account the cluster membership probabilities of the test user $U_t(c_i)$ to compute a weighted sum of the individual cluster contributions from Equation 4.6. Therefore, the final prediction $Pr(U_t, R_t)$ is given by the following formula:

$$Pr(U_t, R_t) = \frac{\sum_{i=1}^{m} U_t(c_i) * Contribution(c_i, R_t)}{\sum_{i=1}^{m} U_t(c_i)} \tag{4.7}$$

Consider the example in Section 4.3.2 again. Suppose we want to predict the rating given by U_3 to R_2. There are two other users (U_2 and U_4), who have reviewed this restaurant. For each of our three clusters, we find the cluster contribution as the weighted sum of the ratings given by these two users to the test restaurant R_2. Using Equation 4.6, and the matrices in Table 4.5 and Table 4.7, we have:

$$\begin{aligned} Contribution(c_1, R_2) &= \frac{\sum_{j=2,4} U_j(c_1) * rating(U_j, R_2)}{\sum_{j=2,4} U_j(c_1)} \\ &= \frac{0.396 * 5 + 0.576 * 2}{0.396 + 0.576} = 3.222 \end{aligned} \tag{4.8}$$

Similarly, $Contribution(c_2, R_2) = 4.793$ and $Contribution(c_3, R_2) = 3.487$ for the other clusters. The final prediction for User U_3 and Restaurant R_2 is computed using Equation 4.7 and the cluster membership probabilities of the test user (U_3) from Table 4.7; given by:

$$\begin{aligned} Pr(U_3, R_2) &= \frac{\sum_{i=1}^{3} U_3(c_i) * Contribution(c_i, R_2)}{\sum_{i=1}^{3} U_3(c_i)} \\ &= \frac{0.38 * 3.222 + 0.502 * 4.793 + 0.118 * 3.487}{0.38 + 0.502 + 0.118} = 4.04 \end{aligned} \tag{4.9}$$

This predicted value is compared with the actual rating given by the user, to compute the error in prediction.

Parameter Selection

The two input parameters to the iIB algorithm are the cluster cardinality parameter M, and the Lagrange parameter β that determines the trade-off between the compression and the relevance

of a clustering. The parameter M needs to be large enough for the data points to be clustered. However, the complexity of the algorithm increases linearly with an increase in the number of clusters. Although, it is possible to run the algorithm offline with periodic updates or to speed up the computation using distributed processing; in our experiments we observed diminishing and unclear improvements in prediction accuracy as the number of clusters increased above $M = 300$. Therefore, for the iIB experiments we fix the number of clusters to 300.

The selection of the trade-off parameter β is more interesting as the prediction accuracy clearly differs with different values for this parameter. For low values of β, implying that the primary focus of the clustering is on the compression of the data, all users are clustered similarly. This makes the weighted restaurant average of the iIB algorithm very similar to the baseline restaurant average of Section 4.2.3. Figure 4.1 shows the percentage improvement of the accuracy of the iIB method using textual features over the accuracy of the restaurant average prediction of Section 4.2.3, for different values of β. The figure represents the accuracy for the task of predicting the star ratings for our three experimental test sets (Section 6.3.1) as β increases from 1 to 30 (with M = 300 clusters). We notice that, initially as β increases, there is a steady improvement in prediction accuracy. However, after $\beta = 20$ there is an increase in the error. This can be explained by the fact that as β increases to very high values, the compression achieved via clustering become irrelevant. This results in poor grouping of users, in turn causing the error values to increase. The clustering for our techniques is done offline and the actual overhead is transparent to the users. All clustering-based recommendation algorithms require this offline step, and it does not impact the actual recommendation time from a user's perspective. An open research direction is to adapt our algorithm to handle incremental data updates without recomputing the entire clustering; this is left for future work.

Clustering based on Full Textual Features

We first experimented with using the iIB method with only the star-ratings matrix converted to the input joint probability $P(X, Y)$. However, as expected the improvements in prediction accuracy over the corresponding results obtained via the rating-based methods (Section 4.3.1), were marginal. The star ratings lack in conveying all the rich information present in the text of the reviews: a user should not be suggested a restaurant, where the overall rating is reflective

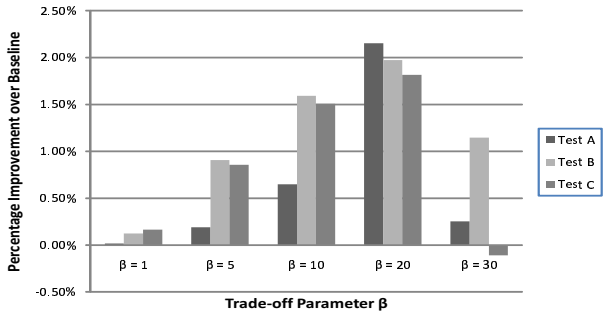

Figure 4.1: Effect of varying β on prediction accuracy.

of topics in which a user is not interested, as illustrated in Example 1. By using the topics and sentiments present in user reviews, we can derive user groupings that take into account the individual interests of users. Therefore, we use the textual information obtained from our classification for clustering users. This allows us to cluster users not only based on the commonality of the restaurants reviewed, but also on their text reviewing patterns or habits.

For the textual content experiments, the input matrix $P(X, Y)$ contains features representing the different sentence types in the text of the review, for each of the 5531 restaurants. In this case, our features mirror the reviewing behaviors of the users, represented by the topics of sentences in the review and the sentiments towards these topics. For the experiment in this section, we used the full textual information derived from the reviews. Therefore, for each restaurant in the data set, we have 34 sentence types representing all combinations of the sentence topics and sentiments (sentences can have a combination of a topic and a sentiment, or one of either), resulting in about 190K features.

Table 4.8 shows the RMSE errors in the predictions for the three test sets when the richer textual information is used as features for the iIB clustering. Note that in all cases, the clustering is done using sentence features, but different ratings (star or text) are used for making predictions. Using textual information for personalized prediction always yields lower error values than the rating-based personalized predictions of Section 4.3.1 (Table 4.4) and the matrix factorization method of Section 4.3.1 (Table 4.4). Moreover, in comparison to the restaurant average-based predictions of Section 4.2.3 (Table 4.2), the improvements in RMSE values shown in the results presented in Table 4.8 are statistically significant ($p - value < 0.05$ using

		TestA	TestB	TestC
Predicting Star Rating	Using Star rating	1.103	1.242	1.106
	Using Textual rating	1.113	1.211	1.046
Predicting Textual rating	Using Star rating	0.692	0.704	0.742
	Using Textual rating	0.544	0.549	0.514

Table 4.8: Prediction RMSE using full textual content for personalized predictions.

the one-sided Wilcoxon test) for all test sets for the task of predicting unknown star ratings using training data star ratings; for the task of predicting star ratings using training data text ratings, the improvements in RMSE values shown in the results presented in Table 4.8 are statistically significant over those of Table 4.2 for the randomly chosen Test sets A and B.

Comparing the personalized predictions based on using coarse rating information (Section 4.3.1) and on using the review text content (Table 4.8) for grouping users, we see that for the traditional recommendation task of predicting unknown star ratings using the training data star ratings, our three test sets A, B and C show a 2.41%, 1.34% and a 1.65% (resp.) improvements when textual information is used.

An important task for a recommendation system is to return the best k product choices for a user. In [66], the author shows that a small improvement in RMSE (even as low as 1%) has a significant impact on the precision of top-k lists. Achieving improvements in prediction accuracy is a notably hard task. The recent Netflix challenge [19], awarded a prize of 1 million dollars to a team achieving a 10% improvement over the existing algorithm; along with step prizes for each 1% improvement. This shows that our methods of incorporating review text in a recommendation system have significant benefits for collaborative filtering systems.

In conclusion, the error values in Table 4.8 show that using the textual information in conjunction with the iIB clustering algorithm improves on the baseline restaurant-average prediction from Table 4.2. Moreover, for our dataset this method is more adept at making personalized prediction than the KNN-based predictions of Section 4.3.1 and the factorization-based method of Section 4.3.1. Thus, our techniques demonstrate that the largely untapped textual information in user reviews contains very rich and detailed information that can be effectively used in a text-based recommendation system to improve rating predictions.

4.4 Qualitative Prediction of Review Components

An important task in understanding and analyzing user reviews is the ability to make fine-grained predictions on the actual content in the reviews. Several websites like TripAdvisor and Yelp have recognized the need for presenting a summary of sentiment towards different product features. Some web sites such as Citysearch, provide binary yes-no answers to questions pertaining to the Ambience and the Service of each restaurant (Romantic? Prompt Seating? Good for Groups?) as well as a numeric Price level. However, this limited summary information is gathered by asking reviewers several yes-or-no questions, making the task of writing reviews very daunting. In addition, the information presented to users is not personalized to match their tastes.

In this section, we describe our techniques for making fine-grained predictions of user sentiments towards the different restaurant aspects, derived automatically from the text of the reviews. First, we cluster users based on their opinions about an individual aspect of the restaurant (Section 4.4.1); such specialized clustering results in neighbors who have the same sentiment towards the particular restaurant aspect. We use the cluster membership probabilities derived from this topic-wise clustering, to predict the importance that a user will assign for each feature and sentiment in his review. We then translate these predictions to binary like/dislike judgments (Section 4.4.2) towards each restaurant feature and evaluate the prediction accuracy in Section 4.4.3.

4.4.1 Clustering based on Restaurant Topics

The average and personalized predictions of Sections 4.2 and Section 4.3 provide an overall predicted rating for a restaurant that does not differentiate on the various restaurant features. Yet, a user might have different sentiments towards a given restaurant: for instance liking the Food and Price but disliking the Service. To accurately predict the sentiment of the user towards each *individual* aspect of the restaurant (Food, Service, Price, Ambience, Anecdotes, and Miscellaneous), we cluster users along six dimensions, using the sentences belonging to each of the six restaurant topics separately. For each user we obtain six sets of neighbors, one for each identified topic.

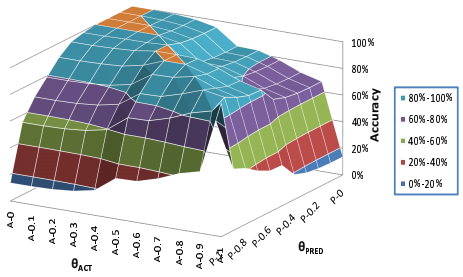

Figure 4.2: Prediction accuracy for positive ambience reviews with varying threshold values.

We cluster users using the information bottleneck method described in Section 4.3.2 with the features belonging to a particular topic. For each restaurant in the data set, we use 5 sentence types features representing all combinations of the sentiments for a particular topic (sentences belonging to a topic can have one of the four sentiments: Positive, Negative, Neutral, or Conflict, or no sentiment), for clustering. The resulting cluster membership probabilities indicate the topic-wise similarity between users; users who have similar sentiment towards the topic across all commonly reviewed restaurants are clustered together. Using the cluster membership probabilities, we now predict the percentage of sentences belonging to each sentence type; not a numeric rating as discussed in Section 4.3.2. The sentence proportion belonging to a particular *(topic, sentiment)* pair is the weighted average of the proportion of sentences of that type written in the other reviews of the restaurant. This weighted average is computed using the prediction algorithm described in Section 4.3.2, where the neighbor ratings are replaced by their sentence type proportions. Therefore, we have predictions for the proportion of sentences of each type that a user may write for the restaurant. In the following section, we describe how these predicted sentence proportions are translated into qualitative binary like/dislike predictions.

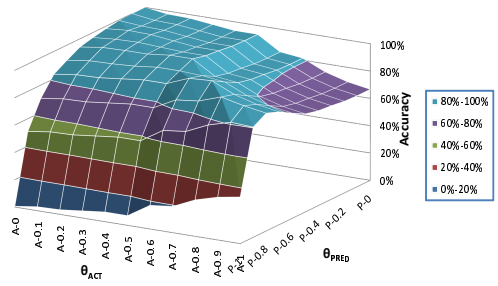

Figure 4.3: Prediction accuracy for negative ambience reviews with varying threshold values.

4.4.2 Topical Sentiment Prediction

We are interested in determining qualitatively whether a user will like (is predicted to have a positive sentiment towards) or dislike (is predicted to have a negative sentiment towards) a particular restaurant aspect. Our data does not contain any ground truth either in the form of binary judgments or ratings for the user sentiment towards the individual restaurant aspects. Therefore, we make sentiment predictions using the predicted proportion of positive and negative sentences belonging to a particular topic (Section 4.4.2). We next learn the ideal parameters for making highly accurate sentiment predictions in Section 4.4.2 and evaluate our predictions accuracy and F1-score.

Parameters for Sentiment Prediction

For each restaurant topic, we need to determine two thresholds: θ_{pred} and θ_{act}. For a topic, if our predicted review composition contains a proportion of positive sentences greater than θ_{pred}, we predict that the user will like this restaurant aspect. Similarly, if our prediction contains a proportion of negative sentences greater than or equal to $(1 - \theta_{pred})$, we predict that the user will dislike the restaurant aspect. Reviews which do not meet either of the conditions above (due to the existence of Neutral and Conflict sentiment sentences) are predicted to be neutral reviews; for such reviews we cannot make polar judgment predictions.

To evaluate our predictions we also need to determine whether the actual review (in the test

set) indicates that the user will like or dislike the particular restaurant aspect. Therefore, for each restaurant topic we define a threshold for the actual user judgment: θ_{act}. The actual judgment towards a restaurant aspect is considered to be positive if the review contains a proportion of positive sentences greater than θ_{act}, if the review contains a proportion of negative sentences greater than or equal to $(1 - \theta_{act})$ the review is considered to be negative, else the review is considered to be actually neutral towards the particular restaurant aspect.

Learning from the Data

We created topic-wise development sets of 215 reviews for each restaurant topic, to empirically determine the threshold values for each topic. Using the training sets, we predicted the review composition for each review set aside in the development sets. We use accuracy as the metric for evaluating our predictions. Accuracy measures the proportion of correctly predicted reviews (true positives and true negatives) to the total number of predictions.

For each restaurant topic, we varied both the actual and predicted parameters. Figure 4.2 shows the accuracy for predicting whether a user will like the ambience in a restaurant. We see that at $(\theta_{act} = 0, \theta_{pred} = 0)$, we predict all reviews to be positive about the ambience and trivially achieve an accuracy of 100%. Fixing $\theta_{act} = 0$, as we increase the prediction threshold θ_{pred}, we predict fewer reviews to be positive on the ambience and the accuracy gradually decreases (true positives decrease and false negatives increase). Similarly fixing $\theta_{pred} = 0$, as we increase the actual threshold θ_{act} the accuracy decreases as true negatives decrease and false positives increase. Interestingly, we get a high prediction accuracy of 95% when we set $(\theta_{act} = 0.8, \theta_{pred} = 0.8)$. This implies that even though we are quite selective in assuming that the review is a positive ambience related review, our prediction methods are able to capture the sentiment with a very high accuracy.

For predicting whether a user will dislike the ambience in a restaurant, the accuracy at varying thresholds is shown in Figure 4.3. Again, we achieve a good accuracy (93%) when we set $(\theta_{act} = 0.8, \theta_{pred} = 0.8)$, as described above. The threshold values set at 0.8 indicate that for a review to be deemed positive on ambience it needs to have more than 80% positive ambience related sentences; whereas if the negative sentences occur only 20% times, the review is deemed negative. This is consistent with our observations of the sentiment distribution [46].

	θ_{act}	θ_{pred}	Combined Accuracy	Positive F1	Negative F1
Food	0.5	0.5	73%	0.85	0.19
Price	0.5	0.5	76%	0.86	0.49
Service	0.5	0.5	61%	0.76	0.22
Ambience	0.5	0.5	76%	0.86	0.49
Anecdotes	0.5	0.5	65%	0.78	0.36
Miscellaneous	0.5	0.5	63%	0.77	0.25

Table 4.9: Evaluating sentiment predictions with $(\theta_{act} = 0.5, \theta_{pred} = 0.5)$.

We have similar trends with varying thresholds for the other 5 restaurant topics, and omit the accuracy plots due to space limitations.

We next evaluate the review sentiment prediction using combined accuracy and F1 scores. The combined accuracy is computed as the proportion of all correct predictions (positive, negative or neutral) to the total number of reviews. Unlike the separate assessment of positive and negative accuracy in the plots above, the combined accuracy is more strict as it does not benefit much from many true negatives. We set the thresholds as $(\theta_{act} = 0.5, \theta_{pred} = 0.5)$, and show the prediction accuracies in Table 4.9. We achieve a good combined accuracy (>73%) for only the Food, Price and Ambience categories. We also include the F1-scores for predicting the positive and negative sentiments. Our results show that we achieve high F1-scores (>76%) for making positive sentiment predictions for all topics, but very low F1-scores for negative predictions.

Fixing the threshold parameters to $(\theta_{act} = 0.5, \theta_{pred} = 0.5)$ is not representative for our corpus. Due to the skew towards positive sentiment in our corpus, for a review to be considered positive on a topic the threshold parameters should be higher than 0.5. Table 4.10 shows the accuracy and F1 scores when the threshold parameters mirror the distribution of positive and negative sentiment towards each topic in our data. A threshold value of 0.8 for the Food category indicates that the positive food related sentences and negative food related sentences have a 80-20 distribution in our classified data. As seen in Table 4.10, learning the threshold values from the text itself, results in a high combined accuracy (>70%) for the main categories of Food, Price, Service and Ambience. Anecdotes and Miscellaneous topics yield lower accuracy values. However, qualitative judgment predictions for these topics do not add much to the user experience. Note that with threshold values learned from the sentiment distribution,

	θ_{act}	θ_{pred}	Combined Accuracy	Positive F1	Negative F1
Food	0.8	0.8	78%	0.87	0.80
Price	0.7	0.7	86%	0.91	0.92
Service	0.7	0.7	70%	0.80	0.72
Ambience	0.8	0.8	85%	0.89	0.87
Anecdotes	0.6	0.6	69%	0.81	0.58
Miscellaneous	0.8	0.8	67%	0.81	0.63

Table 4.10: Evaluating sentiment predictions with threshold parameters learned from the text.

we achieve very high F1 scores for both the positive and the negative sentiments, unlike the results in Table 4.9. A high F1 score for the negative sentiment indicates that our techniques are proficient in detecting the negative judgement in the reviews with high precision and recall; a task that is notably hard due to the lack of sufficient negative examples. Therefore, we set the threshold parameters for the different topics to the values in Table 4.10.

In the following section we discuss an example system that utilizes our text-based rating prediction from Section 4.3.2, as well as the qualitative binary predictions.

4.4.3 Example Interface and Evaluation

Our methods allow us to make rating predictions which indicate the general user assessment of the restaurant, as well as fine-grained qualitative predictions about user sentiment towards individual restaurant features. The key point is that these predictions are made automatically by deriving useful information from the textual content in reviews.

Figure 4.4 shows an example interface for a system built using our techniques. As shown, a user can search for a restaurant and we provide text-based predictions. To evaluate the quantitative rating predictions and the qualitative judgment predictions of such a system, we set aside a new joint-predictions test set containing 30 reviews, and where each user has reviewed at least five restaurants. For the new test set our text-based methods from Section 4.3.2 (text for clustering, as well as textual ratings for predictions) result in a RMSE value of 1.043. For the same 30 test reviews a star rating-based neighborhood model (Section 4.3.1) results in a RMSE error of 1.210. Hence, our text-based techniques show a 13.8% improvement over the star rating-based system. In addition, for this new test set we provide sentiment prediction for the individual restaurant features, using the threshold parameters shown in Table 4.10. The

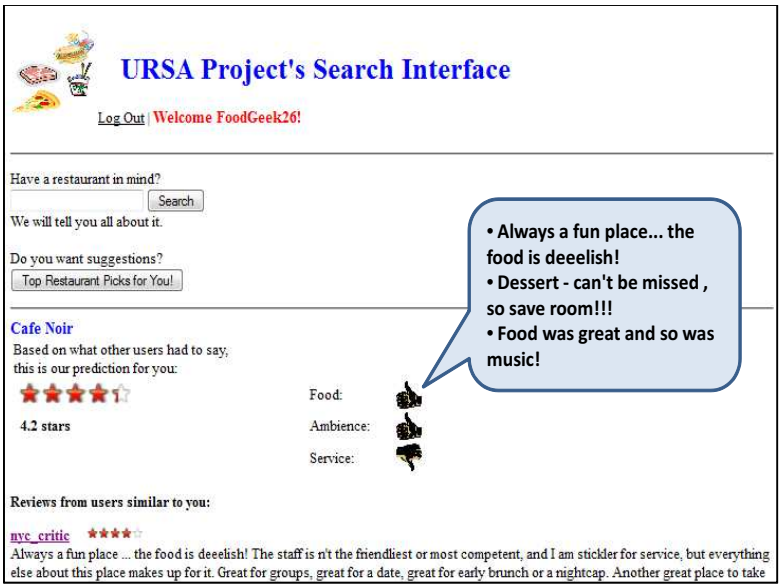

Figure 4.4: Example search interface with rating predictions and fine-grained sentiment predictions.

sentiment predictions have a combined accuracy of 81.8%; indicating that our techniques are proficient in capturing the information present in the review text to make fine-grained personalized predictions.

Our interface also offers an alternate way to accessing the information present in the textual reviews, by providing example sentences belonging to the different *(topic,sentiment)* types. Therefore, a user no longer has to browse through the large amount of unstructured text in the reviews, but can browse a few sentences for each topic and sentiment that reflect the characteristics of the restaurant. Our future work includes choosing the best sentences to be displayed to the user based on length, number of nouns and adjectives, frequently repeating phrases, and other indicators.

Our novel qualitative predictions of individual features is a promising direction to follow to understand and analyze user reviews in detail.

4.5 Conclusions

In this chapter, we presented the user reviews classification and analysis effort performed as part of our URSA project. Our main contribution is the assessment of the impact of text-derived

information in a recommendation system. We show that both topic and sentiment information at the sentence level are useful information to leverage in a review. In addition, we use soft clustering techniques to group like-minded users for personalized recommendations, using the *detailed textual structure and sentiment of reviews*. Our techniques make better ratings predictions using the textual data, and moreover, we make fine-grained predictions of user sentiment towards individual restaurant features.

We are investigating additional refinements to our text-based recommendations, including better text classification strategies and utilizing temporal factors and other available metadata to guide our analysis. In addition, we are interested in the impact of text classification on search over reviews and are implementing tools that allow users to search reviews using topic and sentiment information. Lastly, similar to the study in [29] we are interested in evaluating the performance of our techniques in generating top-k restaurant recommendation lists.

We make our data available at `http://spidr-ursa.rutgers.edu/datasets`, and our code for making personalized predictions using the Information Bottleneck method at `http://spidr-ursa.rutgers.edu/code`.

In the next chapter, we change focus to search over user generated content. As described in Chapter 2, search over user generated forum data has several interesting challenges. We address these by implementing a new search paradigm, allowing retrieval of results at varying focus levels.

Chapter 5

Multi-Granularity Search

Web forums serve as a very popular mean of communication and information exchange. A common approach to gather feedback on a product, disease, or technical problem is to ask a question on an Internet forum and await answers from other participants. Alternatively, one can search through information in forums which often is already present as part of earlier discussions.

Unfortunately, web forums typically offer only very primitive search interfaces that return all posts that match the query keyword. Because of the nature of keyword-based search, short posts containing the query keywords may be ranked high even if they do not have much useful information, while longer posts with relevant information could be pushed down in the result list because their normalized scores are penalized by their size. When issuing queries to forums, users face the daunting task of sifting through a massive number of posts to separate the wheat from the chaff.

As a new search problem, search over forum text yields interesting challenges. Background information is often omitted in posts as it is assumed that readers share the same background knowledge [34]. A critical challenge then for web forums search is to provide results that are as complete as possible and that do not miss some relevant information but that are also focused on the part of individual threads or posts containing relevant text. Therefore, in this type of search the correct level of result granularity is important.

Consider the results in Table 5.1 retrieved in response to the user query *hair loss* in a breast cancer patient forum. Several succinct sentences (A) through (H), shown in boldface, are highly relevant to the query and provide very useful answers. Yet, when a post contains many relevant sentences as in Post1 and Post2, the post is a better result than the sentences alone. Dynamically selecting the best level of focus on the data helps users find relevant answers without having

to read large amounts of irrelevant text. Therefore, our goal is to improve the experience of users searching through web forums by providing results that focus on the parts of the data that best fit their information needs. Our approach allows for search results to be returned at varying granularity levels: single pertinent sentences containing facts, larger passages of descriptive text, or entire discussions relevant to the search topics. In this chapter, we focus our analysis on a patient forum, `breastcancer.org`, although our techniques can be ported to any domain. Our example forum provides very basic search capabilities: keywords are used for filtering posts which are presented chronologically.

We propose a novel multi-granularity search for web forum data to offer the most relevant information snippets to a user query. In particular, we make the following contributions:

- We propose a hierarchical model to represent forum data and present a recursive scoring function over the hierarchy (Section 5.1) which allows us to rank textual objects of varying sizes while taking into account their inherent containment relationships.
- We present a novel score optimization algorithm that efficiently chooses the best k-sized result set while ensuring no overlap between the results (Section 5.2) and show that our optimization algorithm is highly efficient in Section 5.5.
- We study the usefulness of the multi granularity results by conducting user studies to evaluate their relevance (Section 5.5). We show that a mixed granularity set of results is more relevant than results containing only posts, as is the current standard.

This chapter is structured as follows. We discuss our hierarchical data model in Section 5.1, and describe our novel scoring over the multi-granularity hierarchy. We present an efficient algorithm to compute the optimal-scored non-overlapping result set over the multi-granular objects in Section 5.2. In Section 5.3, we describe our forum dataset. We assess the retrieval effectiveness and relevance of results generated by our multi-granularity search in Section 5.5, and demonstrate the efficiency of our score optimization algorithm. We conclude in Section 5.6.

This work was performed as part of the PERSEUS (Patient Emotion and stRucture Search USer interface) project, which aims at helping both patients and health professionals access online patient-authored information by creating tools to process and enhance the textual data in patient forums. The multi-granularity search techniques in this chapter are published in [45].

Example Textual Results	Top-4 Results
Post1: **(A) Aromasin certainly caused my hair loss and the hair started falling 14 days after the chemo.** *However, I bought myself a rather fashionable scarf to hide the baldness. I wear it everyday, even at home.* **(B) Onc was shocked by my hair loss so I guess it is unusual on Aromasin.** *I had no other side effects from Aromasin, no hot flashes, no stomach aches or muscle pains, no headaches or nausea and none of the chemo brain.* Post2: **(C) Probably everyone is sick of the hair loss questions, but I need help with this falling hair.** I had my first cemotherapy on 16th September, so due in one week for the 2nd treatment. **(D) Surely the hair loss can't be starting this fast..or can it?.** *I was running my fingers at the nape of my neck and about five came out in my fingers. Would love to hear from anyone else have AC done (Doxorubicin and Cyclophosphamide) only as I am not due to have the 3rd drug (whatever that is - 12 weekly sessions) after the 4 sessions of AC. Doctor said that different people have different side effects, so I wanted to know what you all went through.* **(E) Have n't noticed hair loss elsewhere, just the top hair and mainly at the back of my neck. (F) I thought the hair would start thining out between 2nd and 3rd treatment, not weeks after the 1st one.** *I have very curly long ringlets past my shoulders and am wondering if it would be better to just cut it short or completely shave it off. I am willing to try anything to make this stop, does anyone have a good recommendation for a shampoo, vitamins or supplements and (sadly) a good wig shop in downtown LA.* Post3: *My suggestion is, don't focus so much on organic. Things can be organic and very unhealthy. I believe it when I read that nothing here is truly organic. They're allowed a certain percentage. I think 5% of the food can not be organic and it still can carry the organic label. What you want is nonprocessed, traditional foods. Food that comes from a farm or a farmer's market. Small farmers are not organic just because it is too much trouble to get the certification. Their produce is probably better than most of the industrial organic stuff.* **(G) Sorry Jennifer, chemotherapy and treatment followed by hair loss is extremely depressing and you cannot prepare enough for falling hair, especially hair in clumps. (H) I am on femara and hair loss is non-stop, I had full head of thick hair.**	**tf*idf** Sent(E) (4.742) Sent (A) (4.711) Sent (C) (4.696) Sent (G) (4.689) **BM25** Sent (D) (10.570) Sent (B) (10.458) Sent (H) (10.362) Sent (E) (10.175) **HScore** Post2 (0.131) Sent (G) (0.093) Post1 (0.092) Sent (H) (0.089)

Table 5.1: Posts and sentences A through H (shown by the boldface text) retrieved for the query *hair loss* in a search over breast cancer patient forums.

5.1 Forum Data Representation

Forum data has an intrinsic hierarchy; usually there are a few broad topics containing several threads on each topic, and each thread contains many posts written by different authors. Effectively searching through forums requires navigating through the hierarchical structure of the multi-level textual objects. We use the natural hierarchy to model forum data, with lower levels containing smaller textual snippets (Section 5.1.1). We then design a unified scoring over objects at all granularity levels (Section 5.1.2).

5.1.1 Hierarchical Data Model

A natural way to look at information in web forums is to break down the pieces of information into the structural components: each *thread* represents a discussion where users interact through individual *posts* which each contain several *sentences*. We use these three levels of information as the searchable objects in our system.

Figure 5.1 shows an example hierarchy over 12 searchable objects: 6 sentences and 4 posts contained in 2 threads. This representation models the containment relationship between the different object levels, while also representing the strength of the association between parent-child nodes. The leaf nodes contain the keywords in user queries, and larger objects are at higher levels. The edges represent containment indicating that the textual content of the child occurs completely in the text of the parent. The edge weight is equal to the number of times a child occurs in the entire text of the parent, i.e., the edge weight represents the association strength between the parent and the child. The default edge weight is 1, a few instances of edge weight 2 are shown in Figure 5.1 where a word repeats in the sentence, or rarely a sentence like “Thanks.” occurs two times in the same post. Note that other granularity of objects could also be considered: *paragraphs* within posts, *groups of threads* on the same topics, or *groups of posts* by an author.

The hierarchical model for forum data allows us to effectively store, access and score the objects at multiple levels by providing efficient access to the parents and children of a node. We now use this data hierarchy to develop a unified scoring function.

5.1.2 Scoring Variable Granularity Objects

Our search system enables users to retrieve results at different levels of granularity: sentences, posts and threads. A challenge of the search is to provide a scoring methodology that assigns comparable scores across all levels of objects and that incorporates the containment between objects. In this section, we first outline the shortcomings of traditional IR scoring for evaluating multi-granularity objects. We then design a novel scoring function that recursively computes scores for the nodes in our data hierarchy.

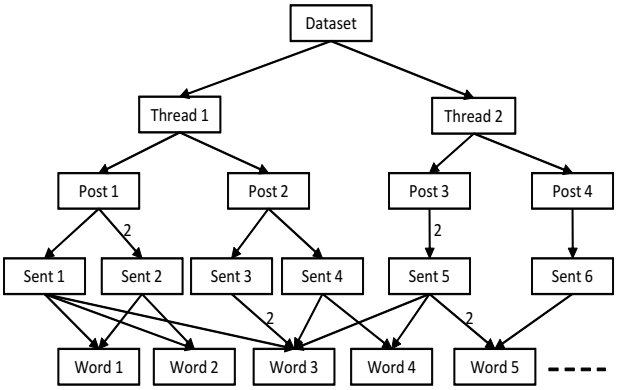

Figure 5.1: Example searchable hierarchy over forum data.

Traditional Scoring Mechanisms

The popular *tf*idf* scoring increases proportional to the frequency of a term in the document, but is offset by the number of documents containing the term. In a multi-granularity system, the documents would include sentences, posts and threads. Suppose a search term occurs only in one sentence in a thread. A basic *tf*idf* scoring will assign the same score to the sentence as the post and thread containing the sentence. This is not ideal as users will have to read the entire thread to find the single relevant sentence.

A common variation of the *tf*idf* scoring is to weight the score of an object with the character length [77], as shown below:

$$Score_{tf*idf}(t,d) = (1 + \log(tf_{t,d})) * \log(\frac{N}{df_t}) * (\frac{1}{CLength(d)^{\alpha}}) \tag{5.1}$$

where the search term is t, the document is d, N is the total number of documents, $tf_{t,d}$ is the

frequency of t in d and df_t is the number of documents containing t. Equation 5.1 correctly assigns higher scores to smaller objects, all other aspects kept equal. However, consider a typical thread with two orders of magnitude bigger $CLength$ than a sentence. For $\alpha = 0.5$ the thread is penalized by an order of magnitude more than the sentence. If the sentence has $tf = m$, for the thread to have a comparable score it needs to have $tf = m^{10}$, which is inadequately large. Hence, Equation 5.1 fails to score objects at multiple levels in a comparable manner.

Another popular IR scoring is Okapi *BM25* [77] which has two parameters: b ($0 \leq b \leq 1$) controls the scaling by document length and $k1$ controls the effect of the term frequency. With the ideal parameter selection (for instance, making the impact of size negligible), the *tf*idf* and *BM25* scoring methods could be tweaked to assign comparable scores to sentences, posts and threads in our corpus. However, these existing methods score textual objects in isolation and ignore the containment relationship amongst the multi-granular objects. A post containing many relevant sentences should have a higher score than the individual sentences, presumably the overall post is more coherent and meaningful. Yet, if all the relevant information is in a single sentence then the parent post must have a lower overall score. Such containment dependencies are not captured by these existing scoring functions.

Consider the performance of the *tf*idf* and *BM25* scoring on the objects in Table 5.1. We also compare our novel *HScore* scoring, described in the following section. For the query *hair loss*, all three scoring functions score and rank the 3 posts and 8 sentences (A) through (H). Post1 and Post2 are highly relevant posts containing many relevant sentences (A) through (F). These posts are more useful for the search task than the sentences alone. Post3 contains a large amount of irrelevant text. Answering the query with the sentences (G) and (H) by themselves saves user time in reading the irrelevant text in the larger Post3. The top-4 ranked results generated by the *tf*idf* and *HScore* functions (size parameter $\alpha = 0.3$), and *BM25* (typical parameters $b = 0.75$, $k1 = 1.2$) are shown in Table 5.1. The ranking reveals that both the existing *tf*idf* and *BM25* functions favor small sized objects and retrieve top-4 results only at the sentence granularity (*BM25* favors sentences with smallest length, while *tf*idf* picks sentences with highest *tf*). These scoring functions rank sentences (A) through (E) higher than the more meaningful Post1 and Post2. Thus, Table 5.1 shows the failure of the existing scoring

functions in generating a mixed granularity result set. In contrast, *HScore* correctly assigns a higher rank to Post1 and Post2, and correctly selects the sentences (G) and (H) in the top-4 results instead of Post3. Moreover, the only scoring that yields mixed granularity objects is *HScore*, described in the next section.

Hierarchical Scoring Function

We use the ideas motivating IR scoring and our hierarchical data model to design a scoring function over the largely varying sized objects in our search system. Our scoring methodology operates in a bottom-up fashion: first keywords at the leaf level are scored and the scores are built incrementally up the hierarchy. The hierarchy allows scoring every object using only its parent and children relations, thereby treating different levels in a comparable manner. Our hierarchical scoring is built on the following intuitions:

- Score of a node is proportional to the score of its children.
- A node having several children or a large sized node, should have its score decreased proportionally to its size.
- If the association between parent and child is strong, as learned from the edge weights, then this child contributes strongly to the overall score of the parent. This is what is commonly captured by the *tf* metric. In Figure 5.1, $Word5$ is a stronger contributor to the score of $Sent5$ than $Word3$ or $Word4$.
- If a child node occurs frequently in the corpus, it carries a lower weight in contributing to each parent's score. $Word3$ is a commonly occurring node and is less influential in affecting the score of its many parents. This is what is commonly captured by *idf*.

Therefore, the score for a node i in our hierarchy with respect to the search term t and having j children is given by the following:

$$
\begin{aligned}
HScore(t,i) &= \sum_{j} \left[\frac{(1+\log(ew_{ij})) * HScore(t,j)}{1+\log(P(j))} * \frac{1}{C(i)^{\alpha}} \right] \\
&\quad \ldots \text{if } i \text{ is a non-leaf node} \\
&= 1 \ldots \text{if } i \text{ is a leaf node containing } t \\
&= 0 \ldots \text{if } i \text{ is a leaf node not containing } t
\end{aligned} \tag{5.2}
$$

where $HScore(t,n)$ represents the score derived using our hierarchical data model for node n w.r.t. term t, ew_{ij} is the edge weight between parent i and child j and is equal to the number of times a child occurs in the entire text of the parent, $P(j)$ is the number of parents of j, $C(i)$ is the number of children of i and the parameter α controls the effect of the size of the node.

To score the forum data at multiple levels, we assume that the $HScore(t,i)$ of a leaf node i containing the query keywords is 1, all other leaf nodes have a score 0. We then recursively compute scores for all parent and ancestor nodes containing the query keywords. For queries with multiple terms, the final score of a node is the sum of its scores for individual terms [77]. Thus, our bottom-up hierarchical scoring leverages the containment amongst the multi-granular objects, which is ignored by existing scoring functions.

Note that the scores computed using the hierarchical scoring function of Equation 5.2 are not monotone to allow for comparable scores across all granularity levels. Having computed the scores of nodes in one level, we cannot guarantee any bounds on the scores of the parents or ancestors. As a result, our scoring cannot be used in conjunction with popular top-k algorithms [36]. Our current implementation scores all objects and retrieves the best k objects using the techniques described in Section 5.2.

Size Weighting Parameter

Our hierarchical scoring function has a size weighting factor inversely proportional to the number of children of the node. The score of a node i having many children, or a large sized node, is penalized by a factor $C(i)^{\alpha}$. Therefore, the parameter α controls the composition of the mixed granularity results.

Figure 5.2 shows the average composition of the top-20 results across 18 queries listed in

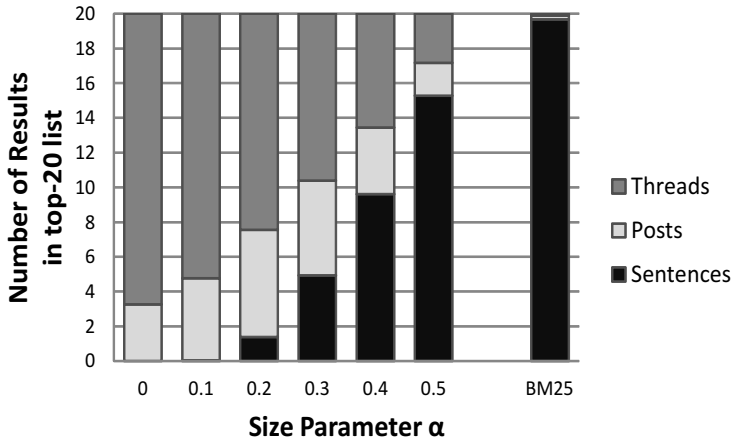

Figure 5.2: Composition of top-20 result sets with varying α in the hierarchical scoring function and *BM25* scoring function.

Table 5.4, for varying values of α. At $\alpha = 0$, the scoring function ignores the size of the objects and larger objects possibly containing many relevant children have higher scores. Hence, at $\alpha = 0, 0.1$ we see a large number of threads in the top-20 results. As α increases we see a mix in result granularity. Eventually at $\alpha \geq 0.4$ the results comprise mainly of smaller objects or sentences, i.e., the size becomes the dominant factor in the scoring causing larger objects to be severely penalized. As discussed in Section 5.1.2, existing scoring mechanisms like *tf*idf* and *BM25* are not suitable for generating mixed granularity results. Yet, as a baseline we show in the rightmost column of Figure 5.2, the composition of the mixed granularity results of the *BM25* scoring. Figure 5.2 shows that *BM25* with typical parameters [77] of $b = 0.75$ and $k1 = 1.2$ retrieves 98% sentences in the top-20 results. Reducing the effect of document length ($b = 0.5$) and increasing the effect of term frequency ($k1 = 2$) still resulted in 88% sentences.

Ideally, we would like to generate a result set with a mix of granularity and hence we set α to 0.2 or 0.3. Note that users can have the flexibility of choosing their preferred result granularity level by varying α. In the following section, we discuss the challenges and strategies for generating a mixed result set for a user query.

5.2 Generating the Result Set

An issue that arises in a mixed granularity search is redundancy. Since the objects at different levels have a containment relationship the same information will be present in a sentence and its parent post and thread. Repetition should be avoided to ensure that users do not see the same information several times [9]. We now describe methods for finding non-overlapping results while ensuring the optimal aggregate score. First, we describe result generation strategies in Section 5.2.1 and then describe our efficient OAKS algorithm for finding the optimal-scored non-overlapping results in Section 5.2.2.

5.2.1 Search Strategies

When generating results with a non-overlapping constraint, it is not sufficient to simply rank objects using their score as computed in Section 5.1. Instead, we need to generate a result set that takes into account the containment relationship among objects. For instance, if we include a post in the final result we should no longer include the sentences in the post or the thread containing the post.

Greedily selecting objects in our hierarchy with the highest scores may result in rejecting other good high scoring objects, causing a decrease in the overall quality of the result set produced. Suppose we want to maximize the sum of the scores of the objects in the result set R. Let us call this optimization function $SumScore$. Over a k-sized result set our optimization problem is as follows:

$$\begin{aligned} \text{Maximize} \quad & SumScore(R) \\ \text{s.t.} \quad & |R| = k, \\ & \forall (x_i, x_j) \in R, x_i \cap x_j = \phi \end{aligned} \tag{5.3}$$

For example, suppose that the searchable objects containing the query keywords are the 12 objects (sentences, posts and threads) from Figure 5.1, and the scores are as shown in Figure 5.3. $Thread1$ has a score of 0.1, $Post1$ has a score of 2.1 and so on. Note that the leaf nodes containing query keywords are used for scoring objects using Equation 5.2, but are not

presented as search results by themselves. We now describe three strategies for generating a result set:

- **Overlap**: Highest scored k objects containing the query keywords are naively included in the result set allowing repetition of text across different granularity levels. From Figure 5.3, for $k = 4$ the result set will contain $\{Post3, Post1, Post2, Sent1\}$ with $SumScore = 8.2$. While the overlap strategy will always return the result set with the maximum $SumScore$, repetition of text should be avoided as is achieved by the next two strategies.

- **Greedy**: Here, we include the highest scored object that is still available in the final result, and repeat this k times. Each time after picking an object we make all its ancestors and descendants unavailable. From Figure 5.3, for $k = 4$, we first pick $Post3$ with the highest score and make $Sent5$ and $Thread2$ unavailable. Repeating this we generate the top-4 greedy result of $\{Post3, Post1, Post2, Sent6\}$ with a $SumScore = 7.0$.

- **Optimal-score**: In general, the greedy strategy may not yield the optimal-score result. From Figure 5.3, the best non-overlapping result set maximizing the $SumScore$ over the $k = 4$ results is $\{Post3, Post2, Sent1, Sent2\}$ with a $SumScore$ of 7.6. This is significantly higher than the $SumScore$ of the greedy method.

Our experiments show that 33% queries have 3 or more overlapping results in just the top-10 results returned by overlap, possibly causing user frustration. Neither optimal-score nor greedy strategies have any overlapping results. In the following section, we describe our novel algorithm for finding the optimal-score result set with no overlap.

5.2.2 Finding Optimal-score Non-overlapping Result

For a result set of size k we wish to maximize a global function like the sum of the scores of the k objects. The problem of finding such a set can be cast as a knapsack problem [103]. Finding a k-sized result set involves solving a knapsack problem with a k knapsack weight limit, unit weight and score as value on all objects, and we have the additional independence constraint amongst the objects. The knapsack problem is NP-Hard and so it is a hard problem to find a

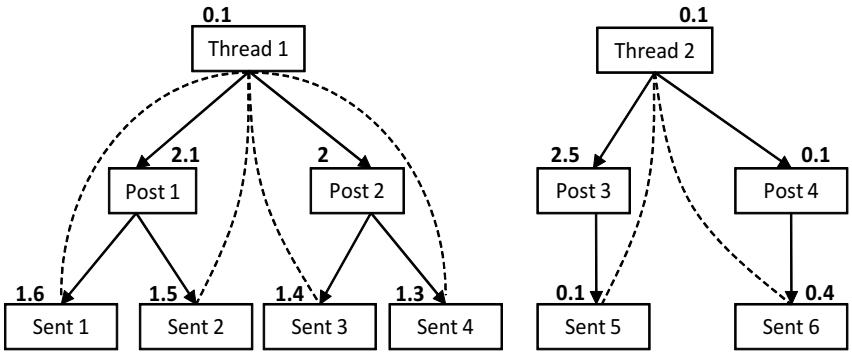

Figure 5.3: Hierarchy of relevant nodes and scores with respect to a query.

non-overlapping optimal-score set. Note that, as described in Section 5.1.2, the hierarchical scoring is not monotone to allow objects at multiple levels to have comparable scores. Hence, top-k optimization algorithms [36] are not useful for generating result sets.

In this section, we describe our approach for finding an efficient solution to this problem. First, we describe our modification to the search graph in Section 5.2.2 and reduce our result generation problem to the independent set problem. In Section 5.2.2, we describe the existing LAIS algorithm for listing all maximal independent sets and then describe our efficient OAKS algorithm for finding the optimal-score result set in Section 5.2.2. For simplicity, throughout this chapter we assume that the optimization task is as described in Equation 5.3. It is trivial to adapt our algorithm to other linear optimization functions, and our experiments with alternate optimization functions did not indicate significant variation in result relevance.

Search Graph Augmentation

We approach the optimization problem of Equation 5.3 as an independent sets problem. An independent set of a graph $G = (V, E)$ is a subset of nodes $V' \subseteq V$ such that each edge in the graph is incident on at most one node in V' [28]. To model the adjacency constraint appropriately, we first augment the search hierarchy with edges connecting all ancestor-descendant pairs shown by the dotted lines in Figure 5.3. To satisfy the non-overlap constraint we now need to find an independent set over this augmented search graph.

In particular, we are interested in finding all maximal independent sets over the search graph. An independent set that is not a subset of any other independent set is called maximal

[59]. Once we find all the maximal independent sets M over the graph, we can generate our optimal-score result set by computing the sum of the scores of the k highest scored elements from each set $m \in M$. However, there are exponentially many maximal independent sets too. We now present an algorithm that generates the maximal independent sets in a specific order with only polynomial delay.

Lexicographic Independent Set Generation

An algorithm for generating all maximal independent sets of a graph in lexicographic ordering is presented in [59]. Assume that we have a fixed ordering over all nodes in the graph. For our problem, we fix the ordering by decreasing object scores. A lexicographic ordering of sets has the first possible node in the first possible set. For instance, for a graph with four nodes ordered as $\{a, b, c, d\}$ having sets as $\{b, c\}, \{a, c, d\}$ and $\{a, d\}$, the lexicographic ordering is $\{a, c, d\}$, $\{a, d\}$, $\{b, c\}$.

The Lexicographic All Independent Sets (LAIS) algorithm [59] is shown in Algorithm 2. The algorithm begins by inserting the first lexicographic maximal independent set in the priority queue Q; this is the set chosen by the greedy strategy. In each iteration, LAIS picks from Q a set S, and then selectively branches from S to find new candidates. For a node $j \notin S$ such that in the sorted ordering j occurs after node $i \in S$, a candidate set C_j is considered. C_j contains nodes in S up to and including j, after removing $\Gamma(j)$ neighbors of j. If C_j is maximally independent on the first j nodes of the sorted order, the first maximal independent set T chosen greedily and containing nodes in C_j is inserted into Q.

Suppose that the LAIS algorithm is applied on the graph shown in Figure 5.3. The fixed ordering of nodes based on the decreasing scores is $Post3$, $Post1$, $Post2$, $Sent1$, $Sent2$, $Sent3$, $Sent4$, $Sent6$, $Sent5$, $Post4$, $Thread1$, $Thread2$, breaking ties arbitrarily. The scores are only used to fix the ordering, and LAIS generates all maximal independent sets irrespective of the scores or sum of scores. The first maximal independent set $S*$ chosen greedily is $\{Post3, Post1, Post2, Sent6\}$. In the first iteration $S = S*$, and we consider all nodes j in the sorted ordering such that j is adjacent to a node $i \in S$, and i precedes j in the sorted order $i \prec j$. The first such node is $Sent1$ and $C_j = \{Post3, Post2, Sent1\}$ is a maximal independent set on the first four nodes of the fixed node ordering. Therefore, we insert the set

$T = \{Post3, Post2, Sent1, Sent2, Sent6\}$ into Q. In this first iteration there exist other j nodes $Sent3$, $Sent5$, $Post4$, $Thread1$, $Thread2$ resulting in five new candidate sets. Setting j to node $Sent2$ or $Sent4$ yields non-maximal sets on the first five or seven nodes respectively, and hence these candidates are not added to Q. In the subsequent iteration, each of the six new sets are popped from Q one at a time for finding new maximal independent sets.

Algorithm 2 Lexicographic All Independent Sets Algorithm

INPUT: Graph G, First maximal independent set $S*$, Priority queue Q

ALGORITHM:

```
Q ← S*
while Q is not empty do
   S = first set from Q
   Output S
   for each node j in G adjacent to a node i in S, and i ≺ j do
      Let S_j = S ∩ {1, ..., j}
      Let C_j = S_j − Γ(j) ∪ j
      if C_j is a maximal independent set of the first j nodes then
         Let T = lexicographic first independent set containing C_j
         Q ← T
   end for
end while
```

The correctness proof and time bounds are in [59]. The delay between outputting sets is bounded by $O(n^3)$. At each iteration a set is extracted from Q, followed by n calculations of new candidates, each time requiring $O(n + m)$ to find a maximal set T. However, LAIS achieves the polynomial delay by using exponential space to store the exponential maximal independent sets in Q.

We now present a new algorithm that improves on these time and space bounds significantly, and finds the optimal-score k-sized result set.

Efficient Result Generation Algorithm - OAKS

We wish to generate an optimal-score result set of size k such that the $SumScore$ of the k nodes is maximized. Finding ordered independent sets is useful to solve this problem. The Optimal-score Algorithm for k Set (OAKS) of results is presented in Algorithm 3 and has some key modifications over LAIS. First, for the k-result set we do not go through all n nodes of the graph to find a maximal independent set. A user is often interested only in the top 10

or 20 results, i.e., $k << n$ (Table 5.4 shows that across 18 queries the average n in our corpus is 23K). Secondly, for a set $S^{(k)}$ picked from the priority queue, we branch only at nodes $j|j \prec End(S^{(k)})$. Since the nodes are ordered by decreasing scores, a node after the k^{th} node of $S^{(k)}$ will have a score lower than or equal to the k^{th} node, and an independent set generated by branching from this node will have a lower overall score. Third, each time we add a k-sized independent set to Q, we check if this set has a higher score than the previous best set. If the k^{th} node of this new higher scored set precedes the k^{th} node of the previously best set, then we reduce the search space further by setting ℓ_{best} = k^{th} node of the new set. We reject sets from the priority queue where the first node of the set $Start(S^{(k)})$ occurs after the current ℓ_{best}.

Algorithm 3 Optimal-score Algorithm for k Set

INPUT: G, First independent set of k nodes $S*^{(k)}$, Priority queue Q

ALGORITHM:

```
Let ℓ_best = node k of S*^(k), sc_best = SumScore(S*^(k))
Q ← S*^(k)
while Q is not empty do
  S^(k)=set in Q,Start(S^(k))=node 1 of S^(k),End(S^(k))=node k of S^(k)
  if Start(S^(k)) ≺ ℓ_best then
    Output S^(k)
    for each node j adjacent to a node i ∈ S, i ≺ j, j ≺ End(S^(k)) do
      Let S_j^(k) = S^(k) ∩ {1, ..., j}
      Let C_j^(k) = S_j^(k) − Γ(j) ∪ j
      if C_j^(k) is a maximal independent set of the first j nodes then
        Let T^(k) = First k-independent set containing C_j^(k)
        Q ← T^(k)
        if SumScore(T^(k)) ≥ sc_best then
          sc_best ← SumScore(T^(k))
          if node k of T^(k) ≺ ℓ_best then
            ℓ_best ← node k of T^(k)
    end for
end while
```

To illustrate the functioning of OAKS let us return to our example graph from Figure 5.3. Using OAKS with $k = 4$, the initial greedy $S*$ is the set $\{Post3, Post1, Post2, Sent6\}$. Note that we only need to find a k-sized independent set. We set sc_{best} to 7.0 and ℓ_{best} to the k^{th} node of $Sent6$. In the first iteration, we evaluate branches of this set only from the nodes occurring before $Sent6$ in the sorted order. Therefore, we do not evaluate branches from

$Sent5$, $Post4$, $Thread1$ or $Thread2$. In the first iteration, new sets of $\{Post3, Post2, Sent1, Sent2\}$ with $SumScore = 7.6$ and $\{Post3, Post1, Sent3, Sent4\}$ with $SumScore = 7.3$ are added to the queue. Thus, after the first iteration only two new sets are entered in Q compared to the six in LAIS. We then further reduce the search space by setting $sc_{best} \leftarrow 7.6$ and $\ell_{best} \leftarrow Sent2$. In just the second iteration, no new candidates are generated and $\{Post3, Post2, Sent1, Sent2\}$ is returned as the optimal-score answer. In comparison with LAIS, the OAKS algorithm evaluates significantly fewer candidates and generates the optimal-score result efficiently.

OAKS Proof of Correctness: We now prove that the k-sized subset returned by OAKS is optimal-scored, i.e., no other k-sized independent set has a $SumScore$ greater than the set returned by OAKS.

Assume a fixed ordering $(v_1, v_2, \ldots, v_n)$ over the n nodes by decreasing scores in response to a query. Let X be the k-sized independent set output by OAKS with nodes $\{x_1, x_2, \ldots, x_k\}$. Our proof consists of two parts. First, we prove that OAKS finds and evaluates at least the first k-sized independent set starting from all nodes v_i appearing before x_k in the sorted ordering, $v_i \prec x_k$, when such nodes are viable, i.e., qualify to be the starting nodes of maximal independent sets. For instance, the nodes $Sent2$ and $Sent4$ in Figure 5.3 do not qualify to be starting nodes of maximal independent sets over the fixed sorted ordering. Secondly, we show that OAKS finds the highest $SumScore$ k-sized set starting with v_i.

Theorem 1. ***Starting nodes of independent sets***

For every node $v_i | v_i \prec x_k$ in the sorted ordering, OAKS evaluates at least the first independent set starting with v_i, when viable.

Proof. OAKS starts with the greedily chosen k-sized independent set with v_1 as the first node. Let us call this set as S with nodes $s_1, s_2, ...s_k$. OAKS then branches from all nodes $v_j \prec s_k$ when such branching yields new maximal independent sets. Some of these branches will result in including a neighbor of s_1 and hence we would reject s_1 in creating the candidate $C_{v_j}^{(k)} = S_{v_j}^{(k)} - \Gamma(v_j) \cup v_j$. This will yield a set with a starting node other than s_1 and we include this set in the priority queue if it passes the maximality test. Repeating this across iterations, we find new candidates with possibly new starting nodes, as long as the starting nodes appears before

the k^{th} node of the current highest $SumScore$ set, i.e., ℓ_{best}.

At some iteration, OAKS finds the optimal-score set X, assigns ℓ_{best} to x_k and inserts X into the priority queue. OAKS arrives at the set X after considering all viable starting nodes appearing before x_1. When OAKS picks X from the queue and starts branching, it evaluates candidates from all nodes before x_k and inserts new candidate sets with possibly new starting nodes into the priority queue. Hence, OAKS evaluates at least the first k-sized independent set starting from all viable starting nodes $v_i \prec x_k$. ■

Theorem 2. ***Optimal-score*** k***-set with*** v_i ***as starting node***

For an independent set with v_i *as the starting node, OAKS finds and evaluates the highest* $SumScore$ k*-sized set starting with* v_i*.*

Proof. From Theorem 1, OAKS inserts in its queue at least the first k-sized independent set with v_i at the starting position when possible. Consider the iteration of OAKS where this first independent set, say F^{v_i}, starting with the node v_i is popped from the priority queue. OAKS finds all branches of F^{v_i} originating from nodes up to the k^{th} node of F^{v_i}, i.e., OAKS branches from all nodes $v_j \prec F_k^{v_i}$. Branching from a node appearing after $F_k^{v_i}$ will result in replacing some nodes appearing before $F_k^{v_i}$ (due to adjacency) with nodes appearing after $F_k^{v_i}$ and such sets will indeed have a lower $SumScore$ than F^{v_i}. Now, for all branches from nodes $v_j \prec F_k^{v_i}$, OAKS finds new candidate sets when $F^{(v_i)} - \Gamma(v_j) \cup v_j$ is maximal on the first j nodes, and inserts these into the priority queue. Repeating this procedure for each of these new candidates $S^{(k)}$, OAKS evaluates all k-sized independent sets by branching from nodes preceding $End(S^{(k)})$. Therefore, across all iterations of OAKS, for a starting node v_i OAKS certainly evaluates the highest $SumScore$ set. ■

For a set Y to have a higher $SumScore$ than X, at least the first node of Y has to appear before the k^{th} node of X, $y_1 \prec x_k$. From Theorem 1, OAKS does not miss evaluating such independent sets with $y_1 \prec x_k$. Moreover, from Theorem 2 we see that OAKS finds the highest $SumScore$ set amongst all k-sized independent sets starting from such a node y_1. Therefore, it follows by contradiction that a Y cannot exist such that it has a higher $SumScore$ than X. OAKS finds the optimal-score k-sized independent set.

In the worst case, OAKS goes through all n nodes to find a k sized subset, i.e., the k^{th} independent node is the same as the n^{th} node. In such a rare scenario, OAKS achieves no performance gain over the LAIS algorithm. However, typically $k << n$. Suppose that the k^{th} node of a set S extracted from Q appears at position p of the sorted order. We find new candidates by at most p branchings from S. Thus, the OAKS algorithm has a time complexity in the order of $O(p^3)$ which gives us a significant improvement when $p << n$, as shown in Section 5.5.3. Moreover, OAKS requires potentially exponential space in p and not in n, and each set in Q has only k nodes, resulting in significant space savings.

Thus, we design the novel OAKS algorithm for efficiently finding the optimal-score k-sized result set. The fixed decreasing score ordering of the sets evaluated by OAKS ensures that in case of a tie in $SumScore$, OAKS outputs the highest scored nodes at the earliest possible position in the ranked list and these results are accessed first by the search user, which is desirable.

5.3 Forum Data Set

We implement a multi-granularity search system on a breast cancer patient forum [57]. The data was collected from the publicly available posts on the online site `breastcancer.org`. The forum data contains threads on a variety of topics useful to breast cancer patients and their families, as well as for health professionals. The search functionality offered by the web site over its forum data is very basic. Results are presented as a chronological list of posts filtered by keywords, irrespective of whether the post is the right level of result for the particular query.

The forum corpus is a large collection of 31,452 threads comprising of 300,951 posts written during the period of May 2004 to September 2010. We used the OpenNLP [3] toolkit for sentence boundary detection and chunking on the text. This resulted in 1.8M unique sentences composed of several search keywords. We prune infrequent mis-spellings and word formulations and retain 46K keywords that occur at least five times in the entire corpus.

We store our data in a MySQL version 5.5 database containing tables for each level in our data hierarchy: words, sentences, posts and threads. Every node in the database contains attributes to store parent and children information, allowing for efficient score computation

using Equation 5.2. For generating a non-overlapping result set as described in Section 5.2, these parent-children relations are extended to find and exclude ancestors and descendants when required. The score computation and result set generation is implemented using Python version 2.6. All experiments were run on a Linux machine (CentOS 5.2) with Intel Xeon (3GHz, 16GB RAM).

5.3.1 Data Characteristics

As described above, the forum data comprises of several posts and threads containing useful information on a variety of topics. In this section, we describe the characteristics of the hierarchy generated over this data while scoring and evaluating the multi-granular objects using our methods.

Table 5.2 shows the composition of the data hierarchy for various settings. First we look at the entire corpus of breast cancer patient forums and build a hierarchy over all the data. We see in the first row of Table 5.2 that the corpus contains 2.3M nodes, and as expected contains a large number (83%) of sentence nodes, 13% posts and 2% thread nodes. These are the result granularities we consider while retrieving search results. The corpus contains 2% word nodes which are used for building the relevant portion of the data hierarchy w.r.t. user query. In addition, the hierarchy over the entire corpus contains 34M edges as shown in Table 5.2 and we see that there is a large number of nodes having multiple parents and children.

Next, we look at the average composition of the nodes across 100 randomly chosen single word queries with corpus frequency in buckets of $freq \leq 50$, $50 < freq \leq 100$, ..., $350 < freq \leq 400$. About 90% of all words in our data have a corpus frequency less than 400. As shown in Table 5.2, for the single word queries the average result granularity is approximately equally distributed between sentences, posts and threads. Hence, our hierarchy does not have an inherent bias of prefering a particular result granularity. The edges in the hierarchies built from single word queries have approximately as many edges as nodes, since we rarely have a situation where a parent node contains many children nodes (in the hierarchy) due to the lack of the overlap in the edges between sentences and words. On the contrary, we also study the hierarchies generated by 2-word and 3-word phrases, which make meaningful search queries. As shown in Table 5.2, the number of nodes in the hiearchy significantly increases as the query

Hierarchy Type	Total Nodes	Sentences	Posts	Threads	Words	Total Edges
Entire corpus	2.3M	1.8M (83%)	300K (13%)	31K (2%)	45K (2%)	34M
1-word $0 <$freq≤ 50	40	14 (35%)	13 (33%)	12 (29%)	1 (3%)	44
1-word $50 <$freq≤ 100	182	64 (35%)	63 (34%)	55 (30%)	1 (1%)	202
1-word $100 <$freq≤ 150	316	113 (36%)	107 (34%)	95 (30%)	1 (0%)	350
1-word $150 <$freq≤ 200	441	152 (35%)	154 (35%)	134 (30%)	1 (0%)	494
1-word $200 <$freq≤ 250	576	199 (35%)	201 (35%)	175 (30%)	1 (0%)	639
1-word $250 <$freq≤ 300	699	250 (36%)	244 (35%)	205 (29%)	1 (0%)	789
1-word $300 <$freq≤ 350	810	293 (36%)	281 (35%)	236 (29%)	1 (0%)	926
1-word $350 <$freq≤ 400	942	333 (35%)	328 (35%)	279 (30%)	1 (0%)	1065
2-word queries	5658	2427 (43%)	2005 (35%)	1223 (22%)	2 (0%)	7604
3-word queries	47245	21550 (46%)	17860 (38%)	7832 (17%)	3 (0%)	57864

Table 5.2: Characteristics of data hierarchy with different query settings.

comprises of multiple words: 5.6K for 2-word queries and 47K for three word queries. Also the number of edges grows due to the overlapping relationships, requiring the need for our hierarchical scoring function built on correctly leveraging the parent-child node relations.

5.4 Other Forum Data

We focus our experiments and evaluation in this chapter on the patient forums described in Section 5.3. While we built our multi-granular search system over breast cancer forums, in this section we describe characteristics of forums from other domains and show that they exhibit similar trends. We did not build and evaluate entire search systems on the different forum domains described here due to budget constraints (as described in Section 5.5 we require manual annotators to assess relevance of search results), and leave analysis on other domains for future work.

Table 5.3 shows the comparison of several data characteristics across three forums: the breast cancer patient forum described in Section 5.3, a corpus containing CNET forum posts (`http://forums.cnet.com`) on the topic of "Computer help", and discussions on Apple forums (`https://discussions.apple.com/`)[1]. As shown in Table 5.3, the three corpora display remarkably similar characteristics. Our breast cancer patient forum is the largest collection with over 31K threads and 301K posts. Yet, threads make up for only 2-3% of all the nodes in the data hierarchy for each of the three forum datasets. A large majority of nodes

[1]Data for CNET and Apple forums was acquired from the DAIS Laboratory at The University of Illinois at Urbana-Champaign (`http://sifaka.cs.uiuc.edu/~wang296/Data/`)

	Breast Cancer Forum	CNET Forum	Apple Discussions
Nodes	2.3M	1M	0.4M
Edges	34M	14M	5M
Avg. node degree	15	14	13
Threads	31K (2%)	29K (3%)	12K (3%)
Posts	300K (13%)	131K (13%)	62K (17%)
Sentences	1.8M (83%)	0.8M (81%)	0.3M (76%)
Words	45K (2%)	30K (3%)	15K (4%)

Table 5.3: Characteristics of data hierarchy across three forums in diverse domains.

are posts and sentences, as expected. Patient forum posts are slightly larger on average, as seen by the large 83% sentence nodes in the patient forums. Overall, all three domains show a comparable proportion on nodes of each type in the data hierarchy as well as the number of edges and average node degree. Hence, our multi-granular search system built over the data hierarchy from each of these three domains will have a similar composition of result granularities, and we can easily port our methods across domains.

Therefore, our domain-independent search methods can be used to enhance user experience in accessing forum data across a variety of topics. In the following section, we evaluate the performance of the hierarchical scoring function and assess relevance of the mixed-granularity search results on this forum data.

5.5 Experimental Evaluation

We now report on our experimental evaluation. In Section 5.5.1, we evaluate the retrieval performance of our hierarchical scoring function from Equation 5.2, by studying the ranks at which target objects are retrieved by our scoring function as compared to the traditional *tf*idf* scoring. We then conduct relevance assessment using crowd-sourcing and show that the relevance of search results improves when users are given a mix of objects at different granularity levels (Section 5.5.2). In Section 5.5.3, we study the optimal-score answer and the efficiency of our OAKS algorithm.

5.5.1 Evaluating Hierarchical Scoring Function

We compare our hierarchical scoring with state-of-the-art *tf*idf* scoring by the ranks at which relevant results are returned, when retrieving objects at a single granularity. We randomly selected 20 queries in the breast cancer domain and a target post that is an *exactly relevant* answer. There might be other posts which are also exact matches for the queries. The target posts however, are highly relevant and a good scoring should retrieve them at lower ranks.

Figure 5.4(a) shows the retrieval performance of *HScore* and *tf*idf* scoring averaged across the 20 test queries, at the granularity of posts. The rank of a target object is normalized by the total number of objects matching the query. A lower normalized rank implies that the target object was retrieved higher up in the list of returned results, which is the desired property. As shown in Figure 5.4(a), both the *HScore* and *tf*idf* scoring retrieve the target posts within the top 3% of results displayed to the users. For all values of α, *HScore* has better retrieval performance than the *tf*idf* scoring at the granularity of posts. As α increases to 0.3 and 0.4, the size of the post dominates the *tf*idf* scoring yielding significantly poorer retrieval ranks as compared to $HScore$. Hence, at the single granularity of posts *HScore* clearly outperforms the *tf*idf* scoring.

Furthermore, we study the effect of the hierarchical scoring at the granularity of threads and sentences. We define target threads as those containing the target posts, and define target sentences as the sentences in the target posts which contain the query keywords and have the best rank. Figure 5.4(b) shows that at $\alpha \geq 0.2$, at the granularity of threads *HScore* outperforms the *tf*idf* scoring as seen by the lower normalized ranks. Figure 5.4(c) shows that both scoring functions perform poorly when retrieving the target sentences with ranks around top 9%. The target sentences are not the best sentence results in response to the queries even though the posts containing these sentences are highly relevant, thus demonstrating the need for dynamically choosing the result granularity level.

Therefore, *HScore* is better than or as good as the *tf*idf* scoring for ranked retrieval over forums, when a single granularity of results is desired. In the following section, we demonstrate the added strength of our hierarchical model which allows retrieving a mix of objects of different granularities. In contrast, *tf*idf* and *BM25* do not provide a unified scoring for objects

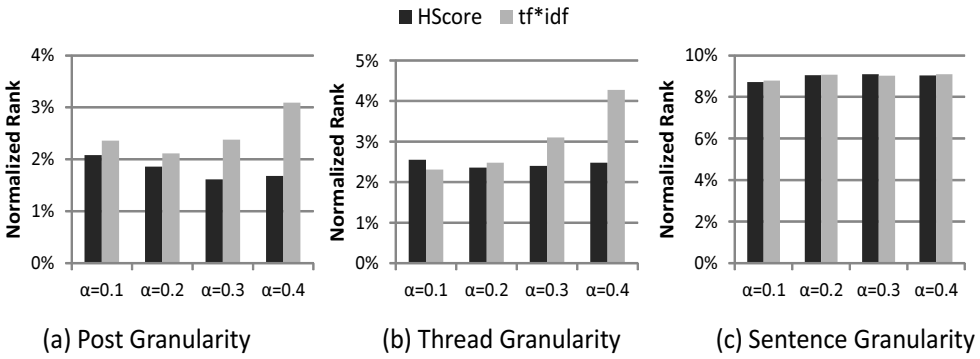

Figure 5.4: Normalized rank of targets at single granularity levels.

at different levels.

5.5.2 Qualitative Evaluation of Relevance

We now evaluate the perceived quality of our results. First, we describe our evaluation setting with details on the relevance assessment scale and quality control mechanisms used for the crowd sourced workers. We then compare the relevance of our multi-granularity results with the traditional posts-only results.

Evaluation Setting

We evaluate our hierarchical data model and scoring function using a set of 18 representative queries chosen randomly. These queries were chosen from different areas of interest for a breast cancer patient from side effects of medicines, alternate treatment options, and food and ingredients beneficial to patients. The queries contain 1 to 3 keywords with an average of 1.8 keywords per query. Table 5.4 lists the queries and the number of searchable nodes at each level of granularity that contain the query keywords. Thus, we need to score and rank a varying number of nodes ranging from a few hundred nodes to 200K nodes (for the query *scarf or wig*).

We compare our mixed granularity search with strategies returning only posts, as current search systems over forums return isolated posts. Relevance judgments for four different experimental search systems were gathered: the *Mixed-Hierarchy* search strategy comprising results at multiple granularity levels scored using our hierarchical scoring function from Equation 5.2 and using our OAKS result generation algorithm, the *Posts-Hierarchy* search system

Query	Sentences	Posts	Threads
broccoli	433	345	229
herceptin side effects	51808	38750	15429
cyst	1006	787	570
emotional meltdown	2354	2175	1643
mole	80	73	62
herbal remedies	648	555	470
accupuncture	196	157	119
medicaid	414	309	186
dangerous pregnancy	1329	1154	944
scarf or wig	102697	73270	22801
tooth sensitivity	1015	844	650
shampoo recommendation	1548	1425	1161
antioxidants	469	324	222
hair loss	23887	15900	6084
organic food	7508	5682	3119
stem cell	4063	2788	2170
hot flash	9244	8114	4317
broccoli sprouts	582	455	288

Table 5.4: Queries for evaluating search systems.

that retrieves results comprising only of *posts* scored using the hierarchical scoring function, the *Posts-tf*idf* search strategy which retrieves posts scored using traditional *tf*idf* scoring, and the *Mixed-BM25* search strategy that uses the *BM25* scoring with the OAKS algorithm to generate results at multiple granularities. As discussed in Section 5.1.2, existing scoring methods like *tf*idf* and *BM25* do not generate a true mixed granularity result as they favor smaller sentences. In addition, these scoring methods do not incorporate the containment relationships amongst objects. Yet, as a baseline we discuss the relevance of the *Mixed-BM25* search which scores and ranks sentences, posts and threads. The *Posts-tf*idf* search mimics existing forum search approaches, which return only posts using IR scoring functions. We now compare these four search systems by conducting relevance assessment experiments.

Graded Relevance Scale

It is common practice in earlier works to use a graded relevance scale [63]. While it is difficult to estimate relevance accurately, the complexity of the problem is multiplied when asking judges to estimate relevance of objects at multiple levels of granularity. In [14], engineers were

asked to assess relevance of large documents containing structured data at multiple levels. For evaluating our search systems, we adapt their relevance scale [63, 84] designed specifically for assessing relevance at multiple granularity. Therefore, we ask judges to annotate search results with one of the following:

- *Exactly relevant*: Highly relevant information at the exact level.
- *Relevant but too broad*: Document contains relevant information, but also includes other irrelevant information.
- *Relevant but too narrow*: Relevant information lacking context.
- *Partial answer*: Partially relevant information.
- *Not relevant*: No relevant information.

This relevance scale captures user assessment towards varying granularity levels as well as the usefulness of the search result.

Gathering Relevance Assessments

We conducted relevance assessment on the Amazon Mechanical Turk crowd-sourcing website [1]. Workers were given 5 results to a query at a time and were asked to mark the relevance according to the proposed scale. Workers were only shown queries and textual results, without discussion on multi-granularity or containment of results. Workers were also provided with examples of search results belonging to each relevance grade.

Our tasks on Mechanical Turk were answered by high-quality workers with 95% or higher acceptance rate. We evaluated batches of tasks to find spammers based on abnormal submissions, for instance when time taken was very low, and blocked these workers. As an additional quality check, each task answered by the workers had a unmarked honeypot question used to assess worker quality. The honey-pot questions were drawn from a pool of questions evaluated by the authors and had the least ambiguity (we often picked irrelevant text to remove the granularity subjectivity). The honey-pot questions were answered correctly by workers who understood the instructions and who were not spammers. After these quality filtering steps, we retained 71% of the relevance annotations, resulting in 7.6 individual assessments for each

	Query = shampoo recommendation			
	α = 0.1	α = 0.2	α = 0.3	α = 0.4
Rank = 1	Rel Broad	Rel Broad	Rel Broad	Partial
2	Rel Broad	Rel Broad	Rel Broad	Partial
3	Rel Broad	Rel Broad	Rel Broad	Partial
4	Rel Broad	Rel Broad	Exactly Rel	Rel Broad
5	Rel Broad	Rel Broad	Exactly Rel	Partial
6	Exactly Rel	Exactly Rel	Rel Narrow	Rel Narrow
7	Rel Broad	Exactly Rel	Rel Narrow	Not Rel
8	Rel Broad	Rel Broad	Not Rel	Partial
9	Rel Broad	Rel Narrow	Rel Broad	Partial
10	Exactly Rel	Rel Narrow	Partial	Rel Narrow
11	Rel Broad	Rel Broad	Exactly Rel	Not Rel
12	Rel Broad	Rel Broad	Exactly Rel	Not Rel
13	Rel Broad	Exactly Rel	Partial	Not Rel
14	Not Rel	Exactly Rel	Rel Narrow	Partial
15	Not Rel	Exactly Rel	Not Rel	Rel Broad
16	Not Rel	Rel Broad	Rel Narrow	Not Rel
17	Exactly Rel	Rel Broad	Exactly Rel	Not Rel
18	Exactly Rel	Exactly Rel	Partial	Partial
19	Not Rel	Rel Broad	Rel Narrow	Not Rel
20	Not Rel	Exactly Rel	Partial	Not Rel

Figure 5.5: Relevance grades of top-20 results with varying α.

search result on average. The relevance assessments were completed by 175 workers, with 114 s required to complete each task on average. For computing the final vote, we used the expectation maximization (EM) algorithm proposed by Dawid and Skene [33] that takes into account the quality of a worker in weighting his vote. Gathering several votes for each task and utilizing these cleaning and pruning methods reduces the error in relevance judgements, and ensures that the relevance estimates obtained are highly reflective of a general user's perception. Moreover, we use identical relevance experiment settings to compare all the alternative search systems.

Figure 5.5 shows the majority relevance assessment of the top-20 results for the query *shampoo recommendation* generated by our *Mixed-Hierarchy* system. At $\alpha = 0.1$, we see that many results have a grade of Relevant but too broad as at low values of α a large number of results are threads (Figure 5.2). On the contrary, at high values of α the result set mainly contains short textual snippets, and many results have the Partial or Not relevant grades. We achieve the highest overall relevance for $\alpha = 0.2, 0.3$. Interestingly, these are the α values with the highest mix in result granularity as shown in Figure 5.2. Thus, we see that the result sets with a high granularity mix also have a high relevance as assessed by workers.

Comparing Search Strategies

We now compare the relevance for the alternative search strategies described in Section 5.5.2. We evaluate the top-20 ranked lists using mean average precision (MAP) [77]. Computing MAP requires binary relevance assessment. For our experiments we assume that if the users annotate a search result as Exactly relevant, Relevant but too broad or too narrow, then the result is relevant. Table 5.5 shows the MAP at different top-k values for the alternative search systems. As described earlier, the composition of the ranked list of results and the overall perceived relevance varies with the size parameter α. Table 5.5 shows that the *Mixed-Hierarchy* search system has a clearly higher MAP than the two posts only systems, for α values of 0.1, 0.2 and 0.3. *Mixed-Hierarchy* performs significantly better for $\alpha = 0.2$ when the top-20 OAKS non-overlapping result set has 2 sentences, 5 posts and 13 threads on average, as well as at $\alpha = 0.3$ with 6 sentence, 5 posts and 9 threads on average; yielding the highest mix in result granularity. The percentage improvement of our *Mixed-Hierarchy* system over *Posts-Hierarchy* at top-20 is 35%, 34% and 16% for $\alpha = 0.1, 0.2, 0.3$ respectively, and the improvements over *Posts-tf*idf* are 31%, 32% and 18% respectively (statistically significant with $p \leq 0.01$ using the one-sided Wilcoxon test). Moreover, *Mixed-Hierarchy* has very high MAP at the top-1 or top-5 results which are most likely to be accessed by users. The *Posts-Hierarchy* system utilizing our scoring from Equation 5.2 has MAP values similar to the traditional *Posts-tf*idf* system; in terms of relevance at the granularity of posts our scoring function performs as well as traditional scoring mechanisms. Lastly, *Mixed-BM25* has a significantly worse MAP than the other search systems. Recall from Section 5.1.2, that the *Mixed-BM25* results mainly contain sentences which often receive a Partial or Not Relevant grade.

As shown in Figure 5.2, $\alpha = 0.4$ skews the results towards sentences, which often are only partially relevant. This explains the degradation of MAP for the *Mixed-Hierarchy* approach at $\alpha = 0.4$, shown in Table 5.5. On the other end of the spectrum, results at $\alpha = 0.1$ are mainly composed of threads which often have a Relevant but too broad assessment. The MAP measure unfortunately, favors relevant results even if they are too broad or too narrow. We further investigate the relevance of our results by taking the gradation of the annotations into account when comparing the search strategies. Discounted cumulative gain (DCG) [30] is a

Search System	**MAP @**	α**=0.1**	α**=0.2**	α**=0.3**	α**=0.4**
Mixed-Hierarchy	1	1.00	1.00	0.94	0.67
	5	0.99	0.99	0.94	0.74
	10	0.98	0.98	0.90	0.70
	20	0.97	0.95	0.85	0.66
Posts-Hierarchy	1	0.78	0.78	0.78	0.72
	5	0.82	0.81	0.80	0.79
	10	0.76	0.75	0.77	0.78
	20	0.72	0.71	0.73	0.75
Posts-tf*idf	1	0.72	0.67	0.72	0.72
	5	0.80	0.77	0.78	0.78
	10	0.76	0.73	0.76	0.76
	20	0.74	0.72	0.72	0.73
	MAP @	**b=0.75, k1=1.2**			
Mixed-BM25	1	0.33			
	5	0.58			
	10	0.55			
	20	0.54			

Table 5.5: MAP relevance values for alternative search systems.

measure for assessing ranked lists with graded relevance. The DCG accumulated at rank k with rel_i indicating the relevance of the result at position i of the ranked list, is computed as $DCG@k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2 i}$.

For our experiments, we translate the five grades of relevance as follows: Exactly relevant has a score of 5, Relevant but too broad and Relevant but too narrow have a score of 4 and 3 respectively (incomplete information is worse than having to read extra text), Partial answers have a score of 2, and Not relevant has a score of 1. Using these relevance scores we computed the DCG for each of the alternative search systems, as shown in Figure 5.6. Figure 5.6 shows the DCG values for the three search systems. We compute DCG at the top 1, 5, 10 and 20 results for size parameter α varying from 0.1 to 0.4. As shown, the DCG values for the *Mixed-Hierarchy* results are generally higher than the two posts only settings, for $\alpha \leq 0.3$. We also see that the *Mixed-Hierarchy* search system has low errors for $\alpha \leq 0.3$, as shown by the error-bars representing one standard deviation.

Normalized discounted cumulative gain (nDCG) is computed by normalizing the DCG@k with the ideal DCG value or IDCG@k. IDCG is the highest achievable DCG of the ideal

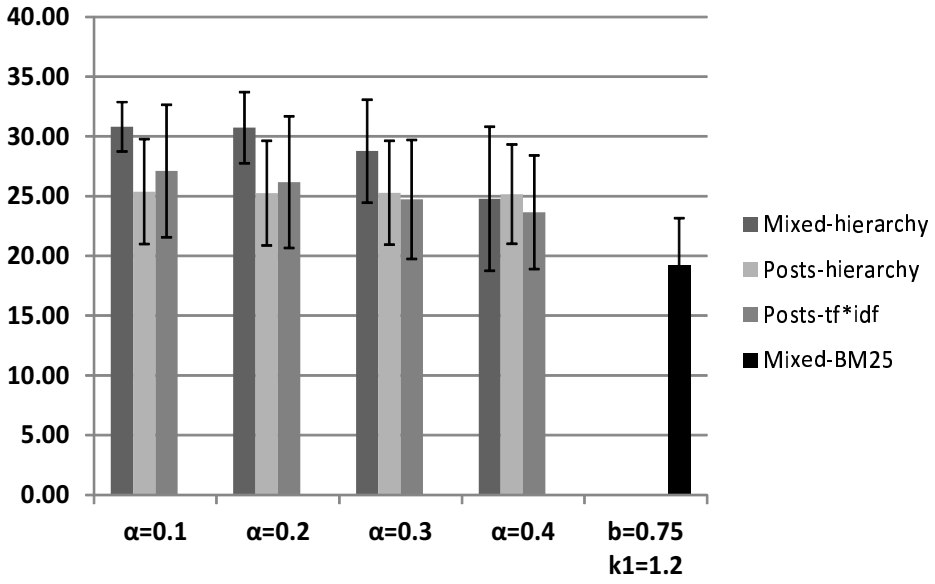

Figure 5.6: Comparison of DCG relevance and standard deviation for the alternate search systems.

ranking. We assume that for each search systems, there exist at least 20 exactly relevant results with the highest relevance grade of 5. Hence, we compute the IDCG at each level k as $IDCG@k = 5 + \sum_{i=2}^{k}(5/\log_2 i)$, and compute the average nDCG across the 18 test queries. Table 5.6 shows the comparison of the nDCG values for the four search systems. Comparing *Posts-Hierarchy* utilizing our hierarchical scoring with the traditional *Posts-tf*idf* system, we see that the overall relevance nDCG of the two systems is very similar. The results show that for $\alpha \leq 0.3$ the *Mixed-Hierarchy* results have a higher nDCG than the two posts only systems, at all of the top-k settings (statistically significant with $p \leq 0.01$ using the one-sided Wilcoxon test). The performance improvement diminishes as α increases to 0.4 when the *Mixed-Hierarchy* results comprise of many partially relevant sentences. Similarly, *Mixed-BM25* results contain many sentences and have a very low nDCG. At $\alpha = 0.2$ with a high mix of results, *Mixed-Hierarchy* clearly retrieves more relevant results at higher positions than the two post only methods. The improvement in nDCG at top-20 by the *Mixed-Hierarchy* system over *Posts-Hierarchy*, *Posts-tf*idf* and *Mixed-BM25* is 22%, 18% and 49% respectively, for $\alpha = 0.2$. These values show that the perceived relevance of a mixed granularity set of results is

Search System	nDCG @	α=0.1	α=0.2	α=0.3	α=0.4
Mixed-Hierarchy	1	0.82	0.83	0.81	0.68
	5	0.82	0.82	0.80	0.69
	10	0.81	0.81	0.76	0.66
	20	0.79	0.79	0.74	0.63
Posts-Hierarchy	1	0.67	0.69	0.70	0.66
	5	0.68	0.68	0.67	0.66
	10	0.65	0.65	0.65	0.64
	20	0.65	0.65	0.65	0.64
Posts-tf*idf	1	0.73	0.67	0.64	0.62
	5	0.72	0.68	0.63	0.60
	10	0.68	0.66	0.62	0.60
	20	0.69	0.67	0.63	0.61
	nDCG @	**b=0.75, k1=1.2**			
Mixed-BM25	1	0.50			
	5	0.53			
	10	0.53			
	20	0.53			

Table 5.6: Relevance nDCG values for the search systems.

higher than post-only results, which is the current state of search over forums. Therefore, there is a clear benefit in generating a mixed granularity result for a user query.

5.5.3 Evaluating Optimal-score Result Generation

In this section, we compare the result set output of the greedy strategy and the OAKS algorithm and evaluate the efficiency of OAKS. As described earlier, experiments were run on a Linux CentOS 5.2 machine with Intel Xeon (3GHz, 16GB RAM).

Comparing Greedy and Optimal-score Strategies

We generated a test bed of 200 single word queries randomly chosen from our corpus. For each of these queries, we evaluated the top-20 results generated using the greedy and optimal-score strategies. Our experiments showed that for 31% of the queries, OAKS returns an answer at top-20 with a higher $SumScore$ than the greedy strategy. Moreover, for 22 of the queries, the improvement occurs within only the top-5 results. Thus, the OAKS algorithm has a noticeable effect on the quality of returned answers. Moreover, OAKS generates the optimal-score result

	Sets Evaluated		Run Time (sec)	
Word Frequency	**LAIS**	**OAKS**	**LAIS**	**OAKS**
20-30	57.6	8.1	0.78	0.12
30-40	102.1	5.1	7.88	0.01
40-50	158.8	5.9	26.94	0.01
50-60	410.2	6.3	82.20	0.02
60-70	716.4	5.3	77.61	0.01
70-80	896.6	8.3	143.33	0.04

Table 5.7: Performance comparison of LAIS and OAKS.

with a low time overhead. Across the 200 queries, OAKS generates the top-20 optimal-score result in 0.09 s on average (greedy required 0.004 s). In comparison, the time required for scoring all objects containing the query keywords was 0.96 s on average. Thus, OAKS yields the optimal-score result set efficiently with a noticeable difference in the few high ranked results.

Comparing OAKS and LAIS Algorithms

LAIS generates all maximal independent sets, while our novel OAKS algorithm evaluates significantly fewer candidates by leveraging the fact that generally users are interested in only the top-k answers to a query, and typically k is much smaller than the size of the search hierarchy. We now compare our OAKS algorithm with the existing LAIS algorithm, both in terms of the run-time and the number of independent sets generated. We randomly generate 100 single word queries in each bucket of search term frequency with buckets of 20-30, 30-40, ... 70-80. In general, search terms in our corpus have much higher frequencies as seen in Table 5.4. We choose these bucket sizes conservatively because LAIS has to evaluate all maximal independent sets which are exponential in the size of the hierarchy. The final result set returned by LAIS and OAKS is the same, however OAKS is significantly more efficient as shown in Table 5.7. We see that as the frequency of the search term increases, LAIS has to evaluate as many as 897 possible candidates for queries with frequency between 70-80. On the other hand the performance of OAKS depends only on k. As before, we set k to top-20 results. As shown in Table 5.7, OAKS evaluates less than 8.3 different candidate sets and finds the optimal-scored answer in less than 0.12 s. Moreover, OAKS achieves this many orders of magnitude speed-up independent of the frequency of the nodes or the data size.

Fanout (θ)	No. of Nodes	Sets Evaluated	Run Time (s)
5	930	8.0	0.02
10	3330	6.6	0.03
20	12630	7.6	0.15
50	76530	6.7	1.13
100	303030	6.7	6.45

Table 5.8: OAKS performance on synthetic data.

OAKS Performance on Synthetic Data

We now generate synthetic datasets to study the efficiency of OAKS on significantly larger search hierarchies than those in our corpus. LAIS cannot compute the optimal-score result on such large datasets due to the exponentially many maximal independent sets. For the synthetic datasets, we vary the number of nodes as well as the dependence between the nodes by varying the fanout θ. We assume three levels in the hierarchy with 30 top-level nodes. Each non-leaf node has θ children. As θ increases the number of nodes in the hierarchy increases greatly, and the dependence between nodes also increases due to increasing number of ancestors and descendants of each node. We assign higher scores to higher level nodes with many descendants, forcing OAKS to evaluate multiple candidates to find the optimal-score k-sized result set. The scores are assigned as follows: leaf nodes have a random real valued score in $(0, 1)$, second level nodes in $(0, 2)$ and top-level nodes in $(0, 3)$.

Table 5.8 shows the performance of OAKS averaged across 1000 runs for each θ. We see that as fanout increases the number of nodes in the hierarchy increases drastically to as many as 303K nodes. Yet, when generating the optimal-scored top-20 result set, OAKS evaluates less than 8 candidates and requires less than 6.45 s. The performance of OAKS depends only on k, and hence we do not see an increase in the number of candidates as θ increases (runtime increases due to the time spent in making increasing number of descendants unavailable). Therefore, OAKS efficiently finds the optimal-score result even under varying data characteristics.

5.6 Conclusions

In this chapter we presented a novel search system over patient forum data performed as part of the PERSEUS project. Our main contribution is the design of a hierarchical scoring methodology that allows several granularities of forum objects to be scored and compared in a unified fashion. Using our scores, we can generate results that contain a mixed set of objects, dynamically selecting the best level of focus on the data. We designed the efficient OAKS algorithm to generate the optimal-scored non-overlapping result set that ensures no redundant information. We conducted user studies to assess the relevance of the retrieved search results and our experiments clearly show that a mixed collection of result granularities yields better relevance scores than post-only results.

In the future, we aim to investigate additional optimization functions for generating multi-granularity results with different properties of the result set. We are currently developing a search interface for representing multi-granularity results in and out of context with visualization tools like highlighting relevant text.

Finding similarities in forum participants can enable a search system to retrieve more useful and relevant results authored by like-minded users. In the next chapter, we develop techniques to capture user similarities across various implicit relations in a unified manner. Such interpersonal similarities allow predicting user connections in the future as well as enhancing keyword search.

Chapter 6

Personalizing Search

Despite their popularity, the search functionality available on the forum sites is often very primitive. The results retrieved in response to a user query usually are posts containing the keywords, ordered by their creation date. There is little or no ranking of results based on the content in the posts. Moreover, isolated posts are not always the right focus level [45]. An under utilized signal in forum data is the information on authorship of posts. Since most online forums require contributors to register, forum participants have an identity and one can leverage the inherent social interactions between forum participants to enhance search. User interactions provide vital clues about their information needs, topics of interest and their preferred other forum participants to answer questions. Some users are very prolific and knowledgeable, and participate in many different discussions on varying topics. Such users are likely to contribute high quality information to forums and their content should have a higher ranking score. Alternately, some users share the same information need, are similar to each other and can benefit from each other. Consider for instance, the study in [57] shows that patients of a particular stage of cancer (Stage I through IV) are more likely to interact with other users with the same progression of the disease. Finding such similar users and weighting their content strongly will enhance the personalized experience of a user. Unfortunately, online forums largely do not provide such personalized search functionality.

Finding similarities in forum participants can enable a search system to retrieve more useful and relevant results authored by like-minded users. Finding such personalized similarity is a complex problem due to the mix of various signals of user affinity. Occasionally, web forums allow users to make explicit friendships, thus providing the social graph over users. Additionally, there exist several implicit cues of user affinity like participating in the same threads or discussing the same topics. Yet, two users having similar information needs at different times

might never participate in the same discussions. For instance, in a forum for mothers several participants will have similar questions about feeding, teething, and sleep patterns. However, some mothers with older children will never participate in newer threads related to infants, even though they have experience raising infants and may have participated in such threads in the past. On the other hand, for a location-based business search, forum participants in the query location are likely to provide answers despite largely varying profiles or topics of interest. Therefore, it is an important and challenging problem to uniformly capture similarities in online users while taking into account multiple implicit signals like user profiles, topics of interest or information needs.

Our approach to address the problem of finding like-minded forum participants is to use a multidimensional random walk that dynamically learns importance of the various inter-user relations. Random walk on graphs, where nodes represent the forum participants and edges represent the strength of node association or node similarity, correctly captures many notions of similarity. However, existing random walk algorithms assume that the underlying graph is homogeneous comprising of nodes and edges of a single type each. Our work extends the random walk (RW) algorithm on a graph to a multidimensional scenario, where each dimension represents a different user relation semantic connecting the nodes. Our algorithm learns the importance of the various interpersonal relations for a user and finds the top-k most similar neighbors across these heterogeneous relations.

In this chapter, we explore the implicit social interactions of forum users to enhance search and personalization of results. In particular, we make the following contributions:

- We design several implicit signals of user affinity and build these user relations over forum participants as discussed in Section 6.1.
- We propose a novel multidimensional random walk algorithm over a heterogeneous graph of user interactions (Section 6.2). Our approach uniformly captures the multiple relations between users with varying weights learned in an egocentric manner, to find the most similar nodes to a user.
- We leverage the multidimensional similarity computation to make predictions on forum

participants who are most likely to answer a question asked by a particular user as described in Section 6.3. We propose this novel prediction task of predicting forum participation which is useful in making recommendations of users and threads to follow.

- Lastly, we enhance keyword search by re-ranking search results using the importance or authority of content contributors (Section 6.4). Our results show that a result list that incorporates such author importance weights has a higher relevance than a purely IR-based text scoring.

The rest of the chapter is structured in the following manner. We describe our forum dataset, and the design of several implicit user affinity signals in Section 6.1. In Section 6.2, we present our multidimensional random walk model for dynamically learning the importance to be associated with the heterogeneous relations between users. We first demonstrate the utility of our multidimensional similarity computation method for enhancing personalized search by link prediction in future forum interactions (Section 6.3). We compare our method with several baselines on homogeneous networks and the existing metapath based approach from [96]. Next, in Section 6.4 we re-rank the results retrieved by traditional *tf*idf* scoring using the importance of the author of the posts as learned through random walks on the multi-relational author graphs. Thus, we incorporate authorship importance in the non-personalized keyword search scenario; a popularly used feature by non-members of forum sites. We conclude in Section 6.5.

6.1 Forum Dataset and Implicit User Relations

We identify and build several implicit connections amongst participants in a breast cancer patient forum [45]. The data was collected from the publicly available posts and discussions on the online site `breastcancer.org`. The forum data contains threads on a variety of topics useful to breast cancer patients such as “Just Diagnosed”, “Relationships, Emotional Crisis, Anxiety, and Depression” or “Healthy Recipes for Everyday Living”. The content in these threads is very valuable for patients and their families, as well as for health professionals to provide the best services.

The search offered by `breastcancer.org` over its forum data is very basic. Results are presented as a chronological list of posts filtered by keywords. The search functionality would

greatly benefit from using author specific information about topics of interest and expertise, in ranking posts presented to a search user.

The forum corpus is a large collection of 31,452 threads comprising of 300,951 posts. The posts in our corpus are written by 15K authors for whom we have unique usernames and an optional signature containing information like location, stage of the disease, date of cancer detection and current treatment plan. We used the OpenNLP [3] toolkit for sentence boundary detection and chunking on the text. We prune infrequent mis-spellings and word formulations and retain 46K keywords that occur at least five times in the entire corpus. We utilize this user-generated text to learn about user interests and expertise.

Our corpus does not contain any explicit social network over the forum participants. We now describe the design of the different relationship graphs linking the forum participants in our scenario and discuss the implications of each.

6.1.1 Thread Co-participation

Our corpus contains 31,452 threads with an average of 9.7 and a median of 7 posts in each thread. The threads provide signals on the interactions between the forum participants authoring the posts in the thread. Usually, participants ask questions or invoke discussions through the first post in a thread, and other participants choose to provide answers or opinions on the topic in the thread. When participants often post in the same threads, it indicates their shared interests or expertise in the topics covered in the threads, or their shared information needs. Therefore, we build a thread co-participation relation between the participants in our 31K threads. For the co-participation matrix, referred to as C, a directed edge exists from a user i to a user j if i posts in a thread after a post authored by j. The edge weight of this link $ew_C(i, j)$ represents the frequency of such interaction and carries a weight equal to the number of unique threads in which i posts after j. A higher edge weight indicates a frequent interaction and a stronger relation between the two users. Note that, C often contains entries in both directions having different edge weights, i.e. $C_{i,j}$ is usually not equal to $C_{j,i}$ due to the asymmetric ordering of posts of user i and user j within threads.

6.1.2 Proximity of Thread Interaction

Threads in forums often span several posts and frequently the theme of discussion changes as participants digress. Often the discussion initiating the thread is forgone and new themes emerge. It is important to capture the proximity or nearness of posts in a thread when developing affinity between forum participants. We build a post separation distance relation matrix D where a directed link from forum participant i to j exists if i posts in a thread after j. The edge weight $ew_D(i, j)$ is computed as the average inverse distance between the posts of user j and user i, averaged across all commonly participated threads. The minimum distance separating two posts is 1 (consecutive posts). As the distance between the posts of i and j increases, the relation is weaker and the edge weight decreases, i.e. $max(ew_D(i, j)) = 1$. Therefore, the relation D captures the closeness of user interactions within threads.

6.1.3 Topical Similarity from Text

The two relations described so far are built on common thread participation. The interaction of participants in the same threads is often determined by temporal factors. Two users going through the same treatment at approximately the same time will interact more often. Yet, it is possible that other forum participants dealt with the same questions and problems, but in the past. Therefore, we now build a relation between forum participants using the similarity in the text in all the posts contributed by them.

We combine all the posts written by a forum participant into a document. Our corpus contains 46K unique words. Building an implicit relation between users using their raw word frequency distributions implies that we do not take into account word synonyms in finding similarities. To take into account semantic similarities between words, we implement a Latent Dirichlet Allocation (LDA) topic model using the Stanford topic modeling toolbox [2] over the documents representing each user. The LDA topic model enables us to derive a probability score for all the users for each topic (we implemented LDA with 100 topics). Thus, users who often write about a topic even with slightly different words in their language model have similar topical probabilities. In addition, user text is now represented with fewer features: 100 probability scores representing coverage of each topic. We then build a text similarity

relation T with a directed link between user i and user j as the cosine similarity [77] of their topical feature vectors. In our scenario, the cosine similarity between two users can range from -1 indicating complete dissimilarity, to 1 indicating perfect similarity. Building a graph with such edge weights will result in edges between every pair of nodes yielding a very dense graph. Moreover, the graph will contain negative edge weights which are not very meaningful in representing user similarity. Hence, we threshold the text similarity and retain only positive edge weight links. Note that, all links in T are symmetric, i.e., $ew_T(i, j) = ew_T(j, i)$.

6.1.4 Profile Similarity

Finally, we build a profile or metadata based relation S between users. Our forum corpus does not contain structured profile information for the participants. We capture this profile information from the optional signatures of authors. 71.3% of posts in our corpus contain a signature from the author. These signatures do not follow specific patterns and are in free-form text format. Users are allowed to write slogans or personal messages in the signatures. Yet, a large majority of users write about their disease stage, treatment options, first diagnosed date and other highly relevant information which we leverage to find user similarity.

For building the signature-based user profile relation S, we first find all unique signatures in our corpus. We then tokenized these to find unigrams, bigrams and trigrams and we retain 10% of the most frequent phrases of each length. This resulted in 11K unique features. Some examples of commonly occurring unigrams were *HER2-, Stage, Grades, 2cm, bilateral, mastectomy* showing the different cancer tumor characteristics and treatment directions. Bigrams included *Stage I* and all other stages of the disease, grade and tumor size information as well as terms like *Diagnosed: 9/10/2009*. Terms like *mastectomy withour reconstruction* were most common in trigrams. Therefore, user signatures contained many relevant terms for finding forum participant similarity based on their disease progression and treatment. We then build links in the user signatures relation S by computing the cosine similarity between n-gram frequencies of pairs of users. Note that all forum participants do not have a signature and these users do not participate in this relation. Users are also allowed to change signatures, and we represent a user by a concatenation of all his unique signatures.

In the following section, we design a novel multidimensional random walk algorithm that

finds similarity between forum participants through a combination of the four similarity indicators C, D, T and S described above, in a unified manner.

6.2 Random Walks for Building User Similarity

Random walks (RW) on graphs are a popular technique to find the important or influential nodes in a graph. Perhaps, the most popular random walk application is the PageRank algorithm [81]. In this section, we first describe the preliminaries of the PageRank algorithm as it is defined on homogeneous networks in Section 6.2.1. We describe the popular Power Iteration method for computing PageRank distribution weights over a graph. The RW computation can be transformed into an egocentric similarity computation using a fixed root node as described in Section 6.2.2. This allows us to capture egocentric affinity scores w.r.t. a fixed node. We illustrate the ability of random walks to capture many different notions of node proximity. Yet, these earlier approaches building on random walks over graphs do not consider different semantic meanings behind user relations. We then describe our novel multidimensional random walk algorithm in Section 6.2.3, and describe how our model can dynamically learn the importance of the various relations and combine these in a weighted transition matrix.

6.2.1 Random Walks on Social Graphs

The PageRank algorithm [81] was developed to determine the importance of a web page in the Internet graph, where the vertices represent the web pages and the edges represent directed hyperlinks. In this section we describe the original PageRank algorithm on homogeneous networks.

Preliminaries

Let $G = (V, E)$ be a homogeneous network with vertices V representing entities of the same type and edges E representing a single relation between the vertices. E contains a directed edge e_{ij} if node i links to node j. In our setting, we assume that all links are not equal and the edge carries a weight ew_{ij} representing the strength of the directed link from node i to node j. For instance, in a social network if a user i repeatedly comments on posts by user j the

weight w_{ij} is high. Let the nodes in the network be numbered from $1, \ldots, n$ and the PageRank of the web pages be represented by the vector P, i.e., $p_1, \ldots, p_n$ are the PageRank scores of the n vertices. The PageRank p_i of a node i is a number in $(0, 1)$ and represents the stationary probability that a random walk reaches this node i.

Iterative PageRank Computation

Let A be a $n \times n$ matrix representing the link structure of the graph G. The entries in A represent the weights of the edges linking nodes. The value A_{ij} is defined to be zero if node j does not link to node i, and $w_{ij}/\sum_k w_{kj} \forall i, j, k \in V$ if node j links to node i. The value A_{ij} represents the probability that the random walk from node j will take the next step to node i.

The PageRank vector P is computed as $P = A \times P$. The PageRank vector P is the eigen vector of the transition matrix A. In experiments, P in each iteration is computed by iteratively multiplying A as shown in Equation 6.1:

$$P^{t+1} = A \times P^t \tag{6.1}$$

The computation in Equation 6.1 is repeated till there is no significant change in P, i.e., $\left\|P^{t+1}\right\|_1 - \left\|P^t\right\|_1 < \epsilon$. At convergence we arrive at the PageRank scores for every node in the network. In practice the relation matrix A is replaced by the transition matrix M which includes adjustment for dangling nodes as well as a teleportation step representing a surfer randomly jumping to any node in the graph.

$$M = \alpha(A + D) + (1 - \alpha)E \tag{6.2}$$

In the above, D is a $n \times n$ matrix representing the transition from a dangling node. For a dangling node j having no out-links, the j-th column of the matrix D has all entries $1/n$ assigning uniform probability of picking any node in the graph. The matrix E represents the teleportation step, i.e., a random surfer gets bored of following out-links from nodes and randomly jumps to any other node in the graph. The matrix E has all entries set to $1/n$. $(1 - \alpha)$ represents the likelihood of teleportation. In short, with a non-zero probability α, a random

walk proceeds along the out-links of nodes, and with a probability $(1-\alpha)$ there is a jump to a random node in the graph. Usually, α is set to 0.85. Thus, the PageRank is computed over this modified transition matrix as $P^{t+1} = M \times P^t$.

6.2.2 Rooted Random Walks

The PageRank computation assigns a score to each node in the graph which represents the node's relative importance. PageRank score for a node in a graph over forum participants is useful to find important users. For personalized search, on the other hand, we are interested in finding similarities between users to find top-k closest neighbors. We now describe a modification of the RW computation that allows us to capture egocentric node similarity, i.e., similarity of nodes w.r.t. a fixed node.

Algorithm

The random walk computation of Section 6.2.1 uses a transition matrix M comprising of the uniform teleportation matrix E as described in Equation 6.2. The rooted random walk (rooted-RW) [72] is computed on a modified teleportation matrix. We fix a node i as the root node of the random walk. The matrix E is modified such that every entry in the i-th row is set to 1 and all other entries are 0. In essence, during the teleportation step the random surfer can jump only to the root node i with a probability proportional to $(1-\alpha)$. Thus, the random walk originating from the root node i periodically resets and returns to i with the probability $(1-\alpha)$. Hence, we are less likely to traverse to distant nodes from i, which is desired since these distant nodes are less likely to be similar to the root node. Let the rooted-RW score [72] of a node j w.r.t. the root node i be $Score(j)_i$, defined as follows:

$$\begin{aligned} Score(j)_i \quad = \quad & \text{Stationary weight of } j \text{ under the RW:} \\ & \text{move to random neighbor with } \alpha \\ & \text{return to } i \text{ with } (1-\alpha) \end{aligned} \tag{6.3}$$

The score $Score(j)_i$ represents the probability of a random walk originating at i and reaching the node j following the links in the graph. In other words, $Score(j)_i$ represents the

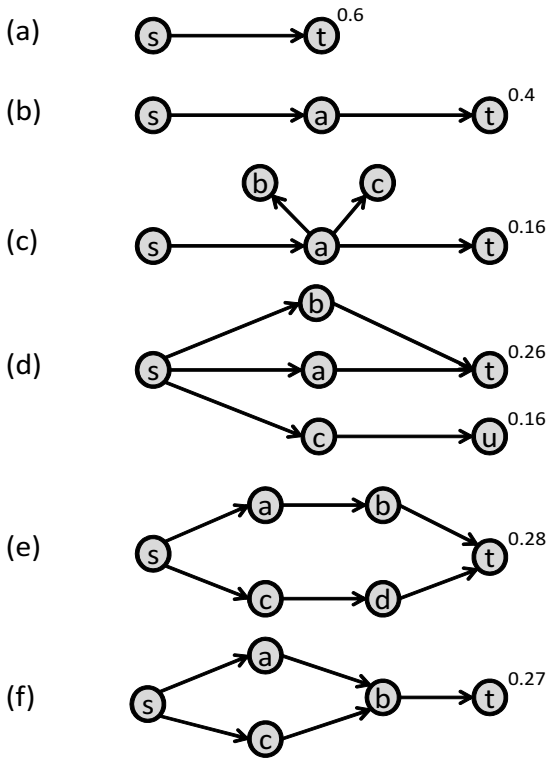

Figure 6.1: Example graphs and node similarity scores captured by the rooted-RW method.

strength of connection or similarity of node j with respect to node i. Therefore, the rooted-RW algorithm provides similarity scores of all nodes in the graph w.r.t. the root node. In the next section, we describe some of the desirable properties of random walks and how they closely capture many notions of node similarity in social networks of users.

Interpreting Node Similarity

Usually, the notion of node similarity depends on the application and context. In a network with entities represented by nodes, the definition of similarity closely depends on the definition of the edges connecting these entities. When the edges represent strength of connection or association, node similarity can be captured using random walks along the edges of a network. We now show how the rooted-RW method described above is apt in capturing many node similarity notions.

Consider the example graphs in Figure 6.1 having the root node set as node s. We compute the scores of target nodes t and u with respect to s using Equation 6.3. All edges have unit

weights. For our application of finding similar nodes in a graph of forum participants, we define node similarity using the following intuitions:

- Nodes closer in the network are more similar. For instance, in a social network with edges representing explicit friendship relations between users, two nodes who are friends are more likely to be similar and are more likely to engage in a shared activity, than two nodes who have a friend of a friend relation. As shown in Figure 6.1(a), the proximity of a target node t w.r.t. the root node s is higher (0.6) with a shorter path connecting the nodes s and t, than in Figure 6.1(b) with a $s-t$ path length of two where $Score(t)_s = 0.4$.

- A path via a popular large out-degree intermediate node contributes a lower weight than a path via a low out-degree node. For instance, in a network representing telephone conversations between pairs of phone numbers, a tele-marketing number is connected to several nodes in the network. Two nodes who are both directly connected to this popular node, are not very likely to be similar. As shown in Figure 6.1(c), if the intermediate node a has a large out-degree, then node t is weakly similar to node s with a score 0.16 as computed using Equation 6.3, which is much lower than $Score(t)_s$ of 0.4 in Figure 6.1(b).

- A node connected to the root node via multiple paths, i.e., through multiple neighbors is more similar than a node connected via fewer paths. As an example, in a social network a user is more likely to befriend a user who is known by many of his current friends. As shown in Figure 6.1(d), the node t has a higher similarity score than the node u because the root node s is connected to node t though more paths. Rooted-RW correctly capture this notion of similarity.

- When two nodes are connected via multiple paths, independent paths indicate a stronger relation than overlapping paths. Consider for instance a communication network with edge weights representing the probability of successfully passing a message. The likelihood of a message from a source node reaching a destination is higher if independent paths connect the two nodes, as there is no single point of failure. Comparing Figure 6.1(e) and Figure 6.1(f), when implementing rooted random walks from node s, the node t in Figure 6.1(e) has a higher similarity score than that in Figure 6.1(f).

the illustrative examples described above show that the rooted random walks are a suitable measure to capture egocentric node similarity. Yet, these walks are defined on homogeneous networks with nodes and edges of a single type each. Often nodes are linked by multiple relations having different semantics. In our context of finding similar forum participants, users are connected due to several reasons, four of which were described in Section 5.3. In the next section, we describe our novel multidimensional random walk algorithm for uniformly capturing node similarity using a combination of several heterogeneous relation types.

6.2.3 Multidimensional Random Walks

In the real world, entities are often linked through different relations. For instance, the likelihood of two people being friends can depend on their shared interests, location proximity, same age or gender and going through similar experiences at the same time. The semantics of these different relations are distinct and merging these to create a homogeneous graph over the connections between people will result in obfuscating some important characteristics. For instance, location proximity might be the dominant reason for making friends for children, while shared interests and hobbies might be more meaningful for adults. Therefore, there is a need to distinguish between the relations and reasons for node similarity, and for dynamically choosing the importance of the relations for each node.

We now present our novel multidimensional random walk algorithm in Section 6.2.3 and show that it uniformly leverages heterogeneous relations for finding node similarity. Next we describe how the multidimensional RW can be parametrized to develop egocentric importance of the different relations in Section 6.2.3. Lastly, we demonstrate the utility of this similarity measure using illustrative examples in Section 6.2.3.

Random Walks on Heterogeneous Graphs

We now formally define heterogeneous graphs and then describe a natural extension of random walks on such heterogeneous graphs.

Definition 3. *A **heterogeneous graph** $G = (V_A, E_R)$ is a graph with a node mapping ϕ and an edge mapping ψ where each node $v \in V_A$ is mapped to one node type $\phi(v) \rightarrow A$ and each*

edge $e \in E_R$ is mapped to a link type $\psi(e) \rightarrow R$. There are A types of objects and R types of links or relations. When $|A| > 1$ or $|R| > 1$, the graph is called a heterogeneous graph [96]. If $|A| = 1$ and $|R| = 1$, then the graph is said to be homogeneous.

For our scenario we are interested in finding the similarity between online users using multiple implicit signals. We have nodes of only one type V_1 and edges of multiple types as developed in Section 5.3. Consider the homogeneous graph of each separate relation linking the forum participants. A graph comprising of a single node type and a single link type can be represented as an $n \times n$ adjacency matrix A, where A_{ij} represents a relation between node i and j with a value proportional to the strength of the relation. In our multi-relational scenario, we have several such matrices $A_1, A_2, \ldots, A_k$ where there exist $k = |R|$ different types of relations linking nodes. A multidimensional random walk is then defined as follows:

Definition 4. *Let $G = (V_1, E_R)$ be a heterogeneous graph with V_1 nodes with R types of links as described above. Let A_1, A_2, $\ldots$, A_k each represent a single relation semantic linking the nodes in V_1. A **multidimensional random walk** is a random walk on the composite adjacency matrix $A = \theta_1 * A_1 + \theta_2 * A_2 + ... + \theta_k * A_k$ where $\sum_i \theta_i = 1$ and all $\theta_i \geq 0$.*

The composite matrix A over the heterogeneous graph is a convex combination of the multiple matrices representing the different semantic relations connecting the nodes in G. In other words, the multidimensional random walk can be interpreted as follows: when a random surfer arrives at a node first a relation is chosen with probability θ_i and then we jump to an adjoining node according to the matrix A_i.

Thus, we have a unified algorithm for combining the different relations connecting online users. We now describe our technique to develop these relation weights θ_i in an egocentric manner.

Egocentric Weights Computation

A critical part of the multidimensional random walk algorithm described above is the computation of the weights θ_i in the combination of the different relations. We define the weights in an egocentric manner, where the importance of a relation is determined by the root node. If the root node has a higher edge weight for links of a particular relation, then this relation is more

significant in finding similarities w.r.t. this root node. For instance, in an online social network, some users might interact more regularly with their friends from school (location-based similarity) while some other users will communicate with people having the same interests. Therefore, personal personal preferences should be taken into consideration while determining weights for the multiple dimensions of relations.

For a root node r, the relation weight θ_i for the i-th relation amongst the $k = |R|$ different user relations is computed as in Equation 6.4 below:

$$\theta_i(r) = \frac{\sum_m ew_{A_i}(r,m)}{\sum_{A_k}\sum_j ew_{A_k}(r,j)} \quad \ldots \forall m \in A_i, \forall j \in A_k \tag{6.4}$$

In the above equation, $ew_i(k,j)$ represents the edge weight or strength of relation between node k and node j in the graph representing the relation i. In other words, the egocentric weights $\theta_i(r)$ to be associated with each relation are developed as the relative weightage of edges of relation type i originating from the root node r to the total weightage of the edges from r. The weight $\theta_i(r)$ is high if a particular relation is more important with respect to the root node r as compared to the other relations. Note that, the weights $\theta_i(r)$ are computed only w.r.t. the root node r and are not updated at each step of the random walk. Thus, we compute relation weights in an egocentric manner taking into account the particular preferences of the root node.

Interpreting Multidimensional Node Similarity

We now demonstrate the utility of our multidimensional random walk algorithm in capturing root-centric similarity by varying the weights to be associated with relations in each dimension. For simplicity, we assume that there are two types of relations represented by the graphs G_1 and G_2 as shown in Figure 6.2(a) and Figure 6.2(b). For each of the example scenarios, we assume that there are two root nodes r_1 and r_2 and we wish to develop a similarity score w.r.t. each of these root nodes. The similarity score over the composite matrix combining the heterogeneous user relations is computed using Equation 6.3.

Figure 6.2(a) shows two graphs G_1 and G_2 representing two different relationship types. When the root node is r_1, the relation represented by graph G_1 has a weight proportional to

$\theta_1 = 3/8$ and the second relation has a weight proportional to $\theta_2 = 5/8$. We expect that the relation in G_2 is more important w.r.t. root node r_1, and the weights correctly capture this. As a result the node similarity scores using multidimensional RWs computed using Equation 6.3 over the combination of two relations, assign the node c with a higher score of 0.096 than the node b (0.072), even though the edge weights $ew_{G_1}(r_1, b)$ and $ew_{G_2}(r_1, c)$ have equal unit weight. In contrast, when we compute scores w.r.t. the root node r_2, the relation in graph G_1 is more important. As a result, the node b has a higher similarity score w.r.t. r_2 than the node c. Hence, Figure 6.2(a) shows how egocentric weights can be developed, resulting in different weights to be associated with the relation types based on the root node. Now looking at Figure 6.2(b), we see that the relation in G_2 is more important w.r.t. root node r_1, and the weights are proportional to $\theta_1 = 6/18$ and $\theta_2 = 12/18$. Therefore, the node r_2 has a higher score ($Score(r_2)_{r_1} = 0.210$) than the node b ($Score(b)_{r_1} = 0.182$), even though these two nodes are connected to r_1 with a total edge weight of 9 units. In addition, w.r.t. the root node r_1 the node c has a lower score than node d as G_2 has a higher relation weight. Lastly, in Figure 6.2(b) when the root node changes to r_2, both relations have an equal weight and w.r.t. r_2 the node c has identical similarity score to node d, as expected.

Therefore, the multidimensional random walk correctly captures notions of egocentric similarity. The weights to be associated with each relation are chosen dynamically based on the root, allowing us to capture the varying importance of the relations for each node in an egocentric manner.

Complexity

Algorithms for finding PageRank broadly use two approaches. The Power Iteration method [81] as described in Section 6.2.1 uses linear algebraic techniques. The time complexity for computing rooted-RW using this method, for one root node is $O(Knd)$ where K is the number of iterations till convergence, n is the number of nodes in a graph and d represents the average neighborhood size. Extending the RW framework to the multidimensional scenario requires computing the composite transition matrix one time for each query root node, as described in Definition 4. The time complexity for computing the composite matrix is $O(nd)$, and we can see that our multidimensional framework does not add a significant overhead to the rooted

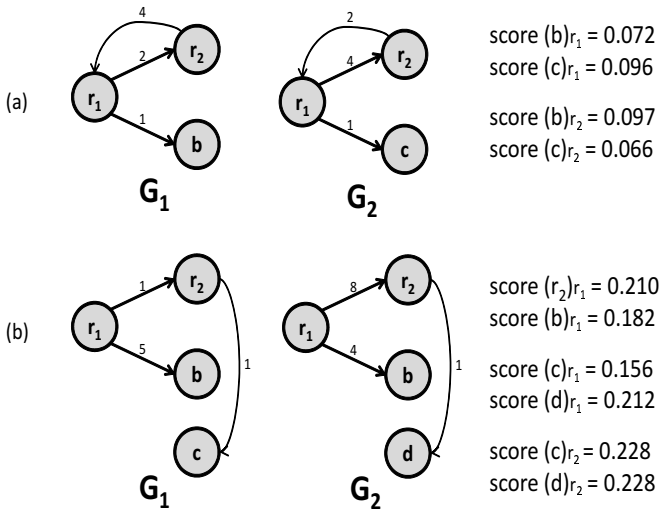

Figure 6.2: Node similarity scores captured by Multidimensional Rooted Random Walks.

RW score computation. The second approach to compute PageRank is based on Monte Carlo approximation which has the benefit of being very efficient and highly scalable [11]. In the future we aim to implement the fast distributed map-reduce based algorithm from [12], which computes approximate rooted-RW scores from each node in the graph in a highly efficient manner.

In the following section, we evaluate the performance of the multidimensional random walk algorithm in predicting future interactions between forum participants using a combination of the four similarity indicators C, D, T and S in a unified manner.

6.3 Personalized Answer Search

When searching for information on online forums, users often pose a question by starting a new thread. Other interested participants then choose to participate in the discussion to help answer the question. An online forum will benefit largely if the likelihood of a user's participation in a thread is known. This will enable users to find and contribute to the best threads, as well as provide the search users with the most useful other users with whom they could interact, become friends and develop meaningful communications.

In this section, we first describe our experimental setting for predicting user participation in

threads in Section 6.3.1. We then use our multidimensional random walks algorithm to find the top-k most similar users to the search user. We use a combination of the various interpersonal relations C, D, T and S described in Section 6.1. Using these relations in Section 6.3.2 we develop a unified similarity score by a rooted multidimensional RW utilizing Equation 6.3. Due to the lack of sophisticated search mechanisms on current online forum sites, users who do not find the required information often tend to repeat similar questions which have already been discussed before. Some forum participants who are experts in the topic and who have answered similar questions in the past are likely to participate in these new threads. In Section 6.3.3 we combine the user similarity scores with the user expertise on the particular question in the first post initiating the thread, to improve predictions on forum participation.

6.3.1 Evaluation Setting

We aim to predict which forum participants are likely to answer a new question posted in a thread. For evaluating our methods, we divide the forum data into a training set comprising of 90% of the threads which were initiated before the remaining threads. These remaining 10% threads are used as a test set. This time-based division is consistent with earlier works [19] where a model is based on the world-view at time t and predictions are made on the new behavior in the next epoch $t + 1$. We have about 2.1K threads in the test set and 28K threads in the training set. Leveraging the information in the training data, we build the defferent adjacency matrices C, D, T and S representing the various relations between the users. The text in the initial posts and the users initiating the test threads are used to predict which other forum participants are most likely to participate in the given discussion. Thus, we design a new prediction task for forum participation which can be used to predict threads or other users which are most meaningful to follow.

6.3.2 Leveraging User Similarity

As described above, we make predictions on the forum participants who are most likely to answer a question posed by the user in the test thread. We do this in the following manner. We first compute the similarity of the user posing the question, called the test user, with all other forum participants. This similarity is developed using Equation 6.3 over each of the four

interpersonal relations C, D, T and S separately using the rooted random walks as described in Section 6.2.2. Therefore, we first compute user similarities using single homogeneous signals of user affinity. We then develop a combined similarity using our multidimensional random walk model which incorporates the four user relations in a unified manner, with the computation of egocentric weights which assign varying importance to each relation. As an additional baseline, we also find the most prolific users in our training corpus and naively predict that these users are most likely to participate in the test threads. In the absence of a mechanism for computing similarity between users, making predictions as most prolific users would be a reasonable approach.

As a baseline for comparison, we also compute the user similarity using the *PathSim* similarity metric defined in [96]. To the best of our knowledge, *PathSim* is the only metric defined on heterogeneous networks. However, *PathSim* has three key differences from our multidimensional random walk model. First, *PathSim* defines a fixed commutation path over the relations, for instance to find users U having similar topical interests T, a path UTU is defined. Although multiple paths of larger lengths can be defined, it is not clear how to choose the best paths or how to combine the similarity computed using different paths. Second, due to the predefined paths, similarity of users separated by a distance longer than the length of the path cannot be computed. Short paths like UTU do not take into account relations like friends of friends. Finally and most importantly, the *PathSim* metric does not allow for computing egocentric importance to be associated with the different inter-user relations, a key beneficial feature of our multidimensional random walk algorithm.

Once we generate the similarity of all users w.r.t. the test user, we rank these users to find top-k most similar users. We predict that these top-k users are most likely to participate in the discussion initiated by the test user. Recall from Section 6.1 that the average number of posts in a thread in our corpus is 30. Hence, we make predictions using small values of k, i.e., $k = 10, 20, \ldots, 100$ most similar users.

Figure 6.3 shows the performance of the different similarity computation methods for predicting forum participation. As shown, our multidimensional RW algorithm has the highest

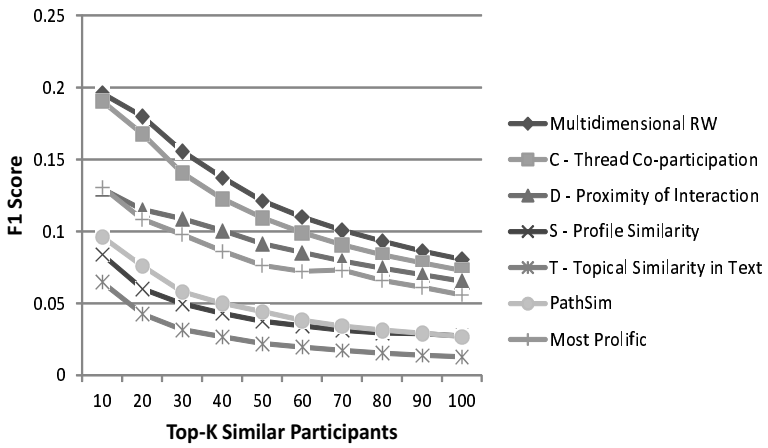

Figure 6.3: F1 score for forum participation prediction using different similarity computation methods.

prediction F1 score amongst all the methods. We see high precision at low values of k neighbors. As k increases precision decreases but recall increases, as expected. The forum participation prediction using single relations has much lower F1 score. The thread co-participation relation C as developed in Section 6.1.1 is the strongest indicator of future interactions between users, but still has a significantly lower F1 than our multidimensional RW approach ($p - value < 0.01$). Our multi-relation approach improves in the prediction F1 score over the thread co-participation relation by 3% at top-10 neighbors and 10% at top 100 neighbors. The naive approach of making predictions of the most prolific users has a significantly worse performance than several similarity-based measures. Therefore, incorporating the different heterogeneous relations in computing user similarity is beneficial in predicting which users are most likely to answer a question posed in a forum thread.

The *PathSim* baseline computation on fixed length paths performs significantly worse than our multidimensional random walk method. Figure 6.3 shows the *PathSim* average prediction performance across all metapaths of length four involving each of the similarity relation once. *PathSim* does not allow for computing egocentric weights, and does not uniformly capture node similarity across the entire graph of relations connecting users. The multidimensional RW method makes more accurate predictions than the *PathSim* method with an improvement of 103% at $k = 10$ and 199% at $k = 100$ neighbors, and these improvements are statistically significant ($p - value < 0.01$ using the one-sided Wilcoxon test).

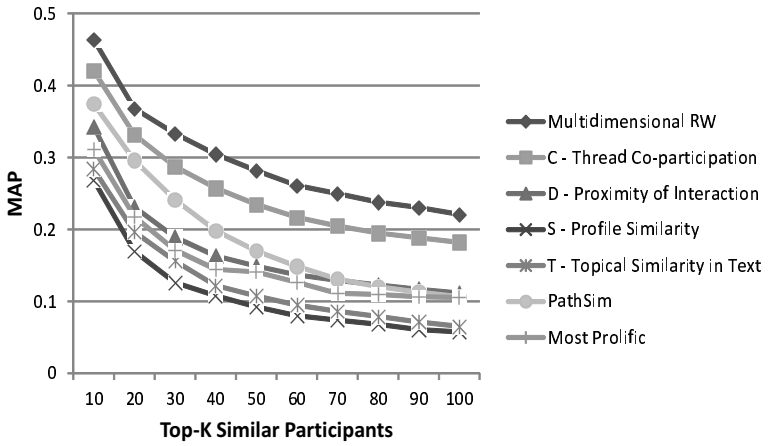

Figure 6.4: MAP for forum participation prediction using different similarity computation methods.

We now compare the alternate prediction methods using Mean Average Precision (MAP). MAP computation takes into account that the correct predictions of forum participation should be the predictions with the most confidence, i.e., the highest similarity with the test user. Figure 6.4 shows the MAP values for predictions using top-k most similar users to the test user. When evaluating MAP, the multidimensional random walk method has a significantly higher prediction MAP than any of the alternate methods. Our multidimensional RW approach significantly improves over the single thread co-participation relation by 10% for $k = 10$ neighbors and 21% for $k = 100$, demonstrating the utility of incorporating multiple relations while computing user similarities. Our method again shows statistically significant ($p - value < 0.01$) improvements over *PathSim* with a 24% improvement at $k = 10$ and a large 108% improvement at $k = 100$.

6.3.3 Leveraging Topical Expertise

In so far, we have generated the similarity between the test user and other forum participants using their relations discovered in the training threads. In addition, we expect that certain users have an expertise or useful knowledge in certain forum topics, as can be learned from their posts in the training data. Consider the situation where a user has encountered a new topic and posts a question on this topic in a forum thread. This test user will not have a high affinity with other users who are experts on this topic in the text-based topical similarity relation T. In such

a scenario, we can improve prediction accuracy by utilizing the topical information in the text of the thread initiating post to find forum participants who have a prior knowledge in the area.

In this section, we combine the user similarity scores developed in Section 6.3.2 using our multidimensional random walk algorithm, with the expertise score of the forum participants w.r.t. the topics in the first post of the test thread. To find the expertise score, we represent each user in our corpus by a concatenation of all the text authored by this user from the training data. Each user is represented by a 46K word vector containing the frequencies of words used in the posts authored by the user. We then use the cosine similarity [77] between the content words in the thread-initiating post and each forum participant. This similarity score allows us to find an expertise score for each user in the topics of the thread-initiating post. Thus, we combine the multidimensional user similarity score *MRWScore* with the topical expertise score *EScore* to generate the final score of a user as follows:

$$UserScore = \beta \times MRWScore + (1 - \beta) \times EScore \tag{6.5}$$

The trade-off parameter β controls the effect of the two components of the score of a user. As $\beta \rightarrow 1$, *MRWScore* dominates the scoring function and we get the same top-k closest neighbors as in Section 6.3.2. β affects the final score of a user and has an impact on the prediction accuracy of forum participants. Table 6.1 shows the prediction MAP for varying top-k users when combining the two user scores using Equation 6.5. Utilizing only the *EScore* at $\beta = 0$ or solely the *MRWScore* at $\beta = 1$ gives lower prediction MAP than the combined method of Equation 6.5, demonstrating the need for a combined method like Equation 6.5. *EScore* alone has a worse performance than our method built on multiple user relations (*MRWScore*). When predicting that top-10 most similar users will participate in the forum threads, our *MRWScore* ($\beta = 1$) shows a 51% improvement in MAP over predictions using *EScore* alone ($\beta = 0$).

As shown in Table 6.1, a combined $UserScore$ shows better prediction MAP than each of the individual scores. We see noticeable improvements when the user expertise *EScore* has a high impact on the overall *UserScore*, as seen at low values of β. These improvements diminish as β increases and the *MRWScore* dominates the overall scoring. For $k = 10$ most similar users, for $\beta = 0.1, 0.2, 0.3$ the percentage improvement over the pure *MRWScore* predictions

Neighbors	$\beta = 0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 1$
Top 5	0.52	0.64 (8%)	0.61 (4%)	0.61 (4%)	0.59
Top 10	0.31	0.50 (8%)	0.49 (5%)	0.47 (2%)	0.46
Top 15	0.24	0.43 (8%)	0.42 (6%)	0.42 (5%)	0.40
Top 20	0.20	0.39 (6%)	0.39 (7%)	0.38 (4%)	0.37

Table 6.1: Prediction MAP and percentage improvements when combining *MRWScore* and *EScore* with trade-off parameter β.

is 8%, 5% and 2% respectively as shown in the parentheses in Table 6.1. Hence, combining the expertise of a user on the topic of the thread has a significant impact in improving prediction accuracy of users likely to participate in the forum.

Therefore, we demonstrate the utility of our multidimensional random walk algorithm for computing user similarity, for predicting users who are most likely to answer questions posed in the new threads. We enable a personalized search that takes into account a users past behavior and interactions to find other similar users and their preferred answers. In the next section, we utilize our multidimensional similarity model to enhance the non-personalized keyword search for a general user of the forum.

6.4 Re-ranking Search Results using Author Information

Users often visit online forums and search using the functionality provided on these web sites. Keyword search refers to such search behavior demonstrated by a random visitor to the forum site, who may or may not have participated in the forum discussions in the past. We cannot assume any information about the search user, and cannot provide a personalized search for this user [1]. Yet, we can leverage the multidimensional similarities between forum participants to find the most influential users in our corpus. Some forum participants are more prolific and write better answers. They participate in many forums on varying topics. Posts written by such users should have a higher importance and a higher rank in the results retrieved for a keyword search. In this section, we discuss our method to generate an authority score for a user and

[1] A search user could be logged into the forum site before issuing a search query. We can then leverage personalized information to improve keyword search. However, our corpus does not contain user session information or query logs. In the future, we wish to combine personalized search with search results re-ranking as described in this section.

utilize this for improving keyword search.

6.4.1 IR Scoring of Posts

Scoring textual documents has been well studied in the Information Retrieval (IR) community. The popular *tf*idf* scoring function and its many variants increase proportional to the number of occurrences of a word in the textual document, but are offset by the frequency of the word in the corpus to account for the fact that commonly occurring words are less important in the overall scoring. A common form of the *tf*idf* function [77] is in Equation 5.1 in the previous chapter, repeated below:

$$Score_{tf*idf}(t,d) = (1+\log(tf_{t,d})) * \log(\frac{N}{df_t}) * (1/CharLength(d)^{\lambda}) \quad (6.6)$$

where the search term is t, the document to be scored is d, N is the total number of documents, $tf_{t,d}$ is the frequency of the term t in d and df_t is the number of documents containing the term. The scoring is inversely proportional to the size of the textual object $CharLength$. This weighting is controlled by a parameter $\lambda, \lambda < 1$. For queries containing multiple terms, the score of a node is the sum of its score for individual terms.

We use this *tf*idf* scoring to retrieve posts in response to a user keyword query. Note that, retrieving results at the granularity of posts might not be the best focus level over the results as described in [45]. Yet, we adhere to this result type as current search systems over online forums retrieve posts in response to a user query. Thus, we score all posts containing the query keywords and refer to this score of a post as its $IRScore_{\lambda}$.

6.4.2 Authority Score of Users

Forum participants demonstrate varying behaviors; some users are more prolific and write many different posts on a wide variety of topics. These users tend to participate in many different threads and interact with many other participants. Posts written by such users are likely to be of higher quality, containing more useful information. To test this hypothesis, we now find the most influential users in our forum data by developing a multidimensional authority score.

For our user authority score computation, we developed a random walk over the multidimensional heterogeneous graph of user similarities, taking into account the four interpersonal relations C, D, T and S from Section 6.1. The composite adjacency matrix from Definition 4 is generated by assigning equal weights θ_i to each user relation A_i. Note that, the different relations have different overall importance in computing the authority scores of users, proportional to the number of edges and edge weights in the different relations. Assigning equal weights θ_i to each relation matrix allows the random walk to take into account the varying importance of relations. We build a random walk over the heterogeneous multidimensional composite matrix in a non-rooted manner, to find the overall importance or influence of the users in our forum corpus, referred to as the $AuthorityScore$ for the users in our corpus. In Section 6.4.4 we use this score to rerank search results retrieved by traditional IR scoring functions, and study the impact and relevance of the new ranking of results.

6.4.3 Qualitative Relevance Evaluation

We now evaluate the perceived quality of our results through crowd-sourced user studies. We first describe our test queries in Section 6.4.3. We conducted user studies to compare the relevance of the returned result sets using the relevance judgment scale in Section 6.4.3. Next, we describe our quality control mechanisms for the crowd sourced workers in Section 6.4.3.

Representative Queries

A critical challenge in studying forum search is the lack of a test set. We evaluate the *tf*idf* scoring of posts using a set of 14 representative queries. These queries were chosen from different areas of interest for a breast cancer patient from side effects of a particular medicine, alternate treatment options, to food and ingredients beneficial to patients. The queries contain 1 to 3 keywords with an average of 1.7 keywords per query.

Evaluating the relevance of all answers to a keyword query is very expensive. Typically users are interested only in the top-k results where k is usually small. We assess the relevance of top-20 results retrieved using the *tf*idf* scoring function for each of the 14 test queries.

Graded Relevance Scale

It is common practice in earlier works to use a graded relevance scale [63]. Posts in forums assume that the same background information is shared by the users. Search results retrieving posts often suffer from the lack of context. For evaluating our ranked list of results, we adapt the relevance scale in [63, 84] designed specifically for assessing relevance at multiple focus levels, taking into account too much or too little context. Therefore, we ask judges to annotate search results with one of the following:

- *Exactly relevant*: Document contains highly relevant information at the exact level.
- *Relevant but too broad*: Document contains relevant information, but also includes other irrelevant information.
- *Relevant but too narrow*: Relevant information accompanied with little context.
- *Partial answer*: Partially relevant information.
- *Not Relevant*: No relevant information.

This relevance scale captures user assessment towards varying granularity levels as well as the usefulness of the search result.

Gathering Relevance Assessments

We conducted relevance assessment on the Amazon Mechanical Turk crowd-sourcing website [1]. Workers were given five results to a query at a time and were asked to mark the relevance according to the proposed scale. Workers were also provided with examples of search results belonging to each relevance grade.

Our tasks on Mechanical Turk were answered by high-quality workers with a 95% or higher acceptance rate. We evaluated batches of tasks to find spammers based on abnormal submissions, for instance when time taken was very low, and blocked these workers. As an additional quality check, each task answered by the workers had an unmarked honeypot question used to assess worker quality. The honey-pot questions were drawn from a pool of questions evaluated by us and had the least ambiguity (we often picked irrelevant text to remove the granularity

subjectivity). The honey-pot questions were answered correctly by workers who understood the instructions and who were not spammers. After these quality filtering steps, we retained 71% of the relevance annotations, resulting in 7.6 individual assessments for each search result on average. The relevance assessments were completed by 175 workers, with 114 s required to complete each task on average. For computing the final vote on the relevance of a result, we used the expectation maximization (EM) algorithm proposed by Dawid and Skene [33] that takes into account the quality of a worker in weighting his vote. Gathering several votes for each task and utilizing these cleaning and pruning methods reduces the error in relevance judgements, and ensures that the relevance estimates obtained are highly reflective of a general user's perception.

6.4.4 Re-ranking results

As described in the previous section, we obtain relevance estimates on a graded scale for the top-20 results for our test queries. We now re-rank the posts retrieved by the *tf*idf* scoring using a trade-off parameter ω to compute a modified score *PostScore* as shown below:

$$PostScore = \omega \times IRScore_{\lambda} + (1 - \omega) \times AuthorityScore \tag{6.7}$$

We compare the relevance of the pure IR scoring with the re-ranked list of results leveraging the user *AuthorityScore*. We evaluate the ranked lists of results using mean average precision (MAP) [77]. Computing MAP requires binary relevance assessment. For our experiments we assume that if the users annotate a search result as Exactly relevant, Relevant but too broad or too narrow, then the result is relevant. Figure 6.5 shows the MAP of the top-10 ranked results for different values of the trade-off parameter ω. As described earlier, the IR scoring returns a different ranked list for each size parameter λ, and we show the MAP for two values, $\lambda = 0.1, 0.2$. As shown in the figure, we get a higher overall MAP when the results are re-ranked using the user $AuthorityScore$ generated by our multidimensional RW over the various implicit user relations. The MAP value peaks in the range of $\omega = 0.7$ to 0.9. Setting ω to 0.9 is a suitable choice (larger focus on IR score) and at this value, the combined $PostScore$ achieves a 5% and 4% improvement over the results ranked using only the IR score for $\lambda = 0.1, 0.2$

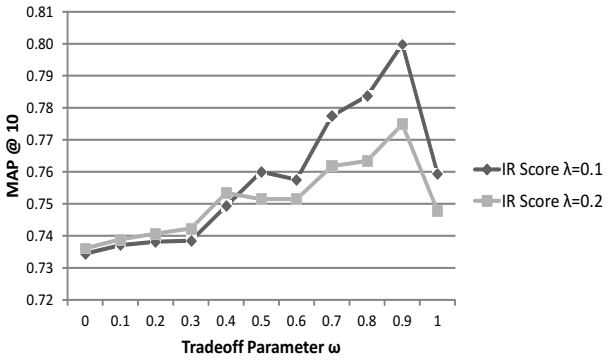

Figure 6.5: MAP of top-10 retrieved results, averaged across the 14 test queries.

respectively. Hence, utilizing the authority score of users can have a noticeable impact on the perceived relevance within as few as the top-10 results.

We now conduct a more fine-grained assessment of relevance estimated by the crowd-sourced users. The MAP measure unfortunately, favors relevant results even if they are too broad or too narrow. We further investigate the quality of the re-ranked results by taking the gradation of the relevance assessments into account when comparing the search strategies. Discounted cumulative gain (DCG) [30] is a measure for assessing ranked lists with graded relevance. DCG takes into account the decrease in importance of results as rank increases. The DCG accumulated at rank k with rel_i indicating the relevance of the result at position i of the ranked list, is computed as follows:

$$DCG@k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2 i} \tag{6.8}$$

For our experiments, we translate the five grades of relevance from Section 5.5.2 as follows: *Exactly relevant* has a score of 5, *Relevant but too broad* and *Relevant but too narrow* has a score of 4 and 3 respectively (incomplete information is worse than having to read extra text), *Partially relevant* has a score of 2, and *Not relevant* has a score of 1. Using these relevance scores we generated the DCG for each of the ranked result lists.

Figure 6.6 shows the comparison of DCG values for the different ranked lists controlled by the trade-off parameter ω. Again we see that the DCG of the re-ranked result set at $\omega = 0.9$ is higher than that of the pure IR scoring; our multidimensional random walk method for computing *AuthorityScore* for forum participants assist in enhancing keyword search result

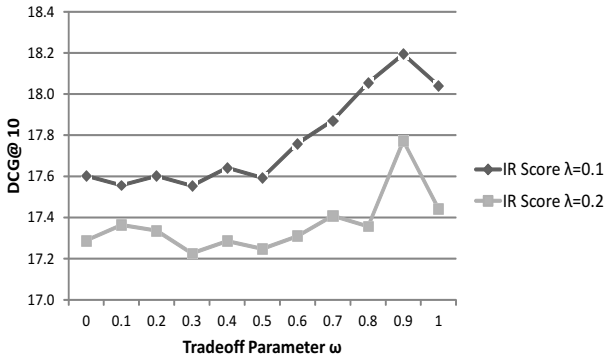

Figure 6.6: DCG of top-10 retrieved results, averaged across the 14 test queries.

relevance.

Therefore, we build several implicit relations between online forum participants and leverage these in a unified manner to enhance personalized and keyword search.

6.5 Conclusions

Online users interact with each other due to a variety of reasons ranging from shared topics of interests, similar demographic information like age, gender and location, or same information need at the same time. In this chapter, we describe a novel multidimensional similarity framework that builds a random walk using heterogeneous relations between users, enabling us to capture user similarity across a variety of reasons in a unified manner. Our heterogeneous framework captures egocentric similarities for a user in our data, and we leverage these similarities to make highly precise predictions on future interactions between users. Finding which users are likely to provide answers to questions posted on a forum improves user search experience in a personalized manner. In addition, we conducted user studies to assess the relevance of search results generated in response to keyword queries. We then enhance keyword search by re-ranking results retrieved by traditional IR scoring by using information on the authority or users contributing to the forums. Our results demonstrate an improvement in overall search result relevance within as few as top-10 results, as perceived by crowd sourced judges. Thus, we uniformly capture multidimensional similarities between users to enhance search and access over online forums.

Chapter 7

Related Work

This chapter reviews and summarizes the literature relevant to the topics covered in this thesis. In Section 7.1 we describe earlier works on capturing topics, sentiments and opinions from textual data. We also review semi-supervised methods like self-training. We then summarize existing work on recommendation systems and algorithms in Section 7.2. In Section 7.3 we describe search techniques over user authored text and describe methods for assessing relevance of search results. Finally, Section 7.4 addresses work on personalization of search and recommendations by using the social network linking online users.

7.1 Identifying Structure over User Generated Text

Identifying both topical and sentiment information in the text of a review is an open research question. Review processing has focused on identifying sentiments, product features [32] or a combination of both [54, 10, 100]. An alternate approach to identifying textual features and sentiments expressed towards them is to use unsupervised classification which has the advantage of not requiring a human-annotated set for training classifiers. In [23], the authors present a unsupervised text classification technique for the Citysearch restaurant reviews data set used in Chapter 4.

Studies like [54] focus on identifying individual product features and sentiments. However, unlike our work in Chapter 4 these studies do not use the extracted opinions and features for collaborative filtering. The approach in [76] identifies aspects or topics by clustering phrases in textual comments, and identifies user sentiment towards these aspects. Most of the work in sentiment analysis operates at the review level. Our processing unit is a sentence, so that a review is modeled as a fine-grained combination of topics and sentiments.

In Chapter 3, we also introduced semi-supervised techniques for identifying topics in the

text, by giving quality guarantees over self-training based approaches. Self-training [106] is one of the oldest and most popular methods for semi-supervised learning. In self-training, a classifier is initially trained on a few labeled examples. Then it is used to predict labels of unlabeled examples, the most confident predictions are added to the training set, the classifier is retrained, and this is repeated until a stopping criterion is met. Self-training is very common in natural language processing, and was applied to various problems, such as named-entity [35, 82] and relation [22, 7, 83] extraction.

The disadvantage of self-training is that it is subject to local optima and does not provide guarantees on the quality of the approximation [110]. Our algorithm for learning ε-subgraphs (Algorithm 1) can be viewed as a type of self-training. Similarly to self-training, the method is iterative and easy to implement. Unlike self-training, we provide guarantees on the quality of the solution.

7.2 Recommendation Systems over User Data

Online reviews are a useful resource for tapping into the vibe of the customers. Accessing and searching text reviews, however, is often frustrating when users only have a vague idea of the product or its features and they need a recommendation. The design of a good recommender system has been the focus of many previous work; a good survey of the work done in this area and the comparison of several techniques is found in [52] and [21]. Recently, the Netflix challenge [19] has brought a lot of attention to collaborative filtering and recommendation systems. The Netflix data as well as the data typically used in other projects on recommendation systems like the pioneer GroupLens project [88], consists of highly structured metadata, often only the rating given by a user to a product. In contrast, our work considers the textual content of reviews to make predictions.

With the advent of online user generated content, social networks and online shopping, recommendation systems have seen a surge in popularity. The recent work by Wang and Blei [104], uses matrix factorization for making predictions for previously rated items as well as items that have never been rated (cold start). Similar to our work, the authors use topic modeling to capture user preferences. In [41], the authors enhance a matrix factorization-based

recommendation system by mapping user or item attributes to the latent factors to make predictions for new users or new items. In [67], the winners of the popular Netflix Challenge demonstrate the effectiveness of matrix factorization techniques in making accurate recommendations, and claim that latent factor models often outperform neighborhood based models. However, our results in Section 4.3.1 show that matrix factorization does not reduce prediction errors for our sparse dataset. In fact, several recent studies like [55] demonstrate the effectiveness of an ensemble or a blend of several individual techniques, and show that ensemble-based methods outperform any single algorithm. Our soft clustering-based models in Section 4.3.2 can be used effectively in such ensembles, and wish to explore this in the future.

The recent work by Leung, Chan and Chung [64] incorporates review text analysis in a collaborative filtering system. While the authors identify features, they unfortunately do not describe their methods and do not summarize all their features or roles. Additionally, the evaluation of their recommendation is done by predicting a 2-point or a 3-point rating. We predict ratings at a fine-grained 5-point rating scale, commonly used in popular online reviewing systems.

The approach in [76] identifies aspects or topics by clustering phrases in textual comments, and identifies user sentiment towards these aspects. However, their techniques often result in finding noisy aspects. In addition, the aspect clustering precision and recall (0.59, 0.64) for their experiments is lower than the average topic classification precision and recall (0.70, 0.64) for our sentence topical classification (Table 3.1). The study in [76] makes a aspect rating prediction and combines these ratings to make a prediction on the overall rating in the review. However, the predictions do not utilize the ratings of similar users to the product, therefore ignoring the social impact of other users on the user assessment of a product. Our soft clustering method from Section 4.3.2 groups users according to their similarities of reviewing behavior, and hence captures the underlying inter-dependencies between user ratings.

7.3 Search over User Data

The standard web search retrieval model returns ten links with summary snippets. Several studies have focused on effectively generating the most readable and appropriate snippets [61,

62]. Recently, researchers have enhanced search results by presenting top ranking sentences and thumbnails along with the links [60], clustering search results [51] and presenting them in a hierarchy of topics [39]. As shown in [60], presenting top ranking sentences along with the web pages enhances user experience. Little work has been done in dynamically choosing the right focus level for the information. Our work on multi-granularity search in Chapter 5 focuses on retrieving text at the appropriate level of granularity to satisfy user needs without the burden of sifting through entire documents, when possible.

Online forums contain rich unstructured data with information on a variety of topics. In [94], the authors use trained classifiers to retrieve sentences about symptoms and medications from patient forums. Such topical analysis of the content posted by users along with the social network of interactions has been successfully used to predict the cancer stage of patients [57]. Textual content has been successfully used to introduce links between different threads in user forums [105]. Yet, very little research has focused on improving search over forums. Models have been developed to incorporate information outside of a post [34], from the thread or the entire corpus, to enhance ranking of the retrieved posts. In [18], the authors use several document homogeneity measures to incorporate more or less information from the document at the passage level. However, the retrieved results by these previous methods are still posts which often suffer from the lack of context. By varying the granularity of search results and by allowing a dynamic mix in search result focus, our methods explicitly incorporate relevant neighborhood context.

Searching through XML documents at different levels of granularity has been well studied and [9] has an overview on XML search and the INEX workshops. We represent the containment relationships of objects at multiple granularities in a hierarchy for computing a bottom-up score. However, our objects are unstructured free-form text objects and relationships specific to XML nodes and attributes do not apply. Avoiding overlapping results in XML has been addressed in previous work by greedily selecting high scored nodes in a path [90], adjusting parent and children node scores after retrieving objects [27] or maximizing utility functions built on object scores and sizes [79]. In contrast, our OAKS algorithm from Section 5.2.2 generates a top-k result set by optimizing a global function, and does not rank objects in isolation. Generating a non-overlapping result set in our scenario implies avoiding repetition of the exact

text within the multi-granular objects. Our work is orthogonal to result diversification [108], where the problem is to find top-k search results that cover diverse topics or sub-topics from a concept hierarchy. Our data hierarchy represents containment of objects and not a topic hierarchy. Previous work in [85, 87] shows that users prefer documents at a medium granularity to judge relevance, rather than short titles and metadata or the entire documents, suggesting that a balance must be struck between too coarse and too fine granularity. However, forum data has little explicit structure and multi-granularity retrieval over such text has not been studied before.

Estimating relevance of textual results is a notably hard task because of the subjective assessments affected by the perception and biases of the judges. There has been a large body of literature with information on the factors affecting relevance estimation as well as a variety of scales and measures for assessing relevance. In [53], the authors suggest eighty factors that affect relevance, from personal biases, diversity of content, browsability and the type of relevance scale. Our approach to mitigating the effect of individual subjectiveness in relevance estimation is to conduct large-scale user studies using crowd-sourcing, and effectively reducing spam annotations, as discussed in Section 5.5.2. As shown in [8], crowd-sourced relevance assessment of XML search obtained via Mechanical Turk had comparable quality to INEX specialized judges.

Patient posts constitute an exciting area for new research: the language is both emotional and technical, the style is often narrative, and forums are highly interactive. In the future, we are interested in studying social interactions in forums.

7.4 Personalization with User Similarities and Preferences

Many studies have discussed the different relations between online users ranging from some early works like [6], where the authors study the connections between users in two diverse social networks and use relations ranging from physical proximity, organizational hierarchy and profile information like gender or age. More recently, the authors in [25] studied user similarity through explicit friendships or relations, through implicit co-participation and engagement with tags and comments and a topic-based similarity computed using terms associated with the

user. These earlier studies indicate that there are many explicit and implicit reasons for user interactions in online communities, and there is a need for a unified framework for combining these diverse signals. Yet, previous works lack methods for such a unified mechanism. In our work in Chapter 6, we use a multidimensional random walk algorithm for addressing this need.

The PageRank citation ranking [81] was the original algorithm used by the Google search engine for finding authority pages on the Web using hyperlinks. There have been several studies that use the random walk methodology for finding authority nodes in a graph including the topic-sensitive page rank computation [50] and personalized PageRank for searches in ER graphs [56]. These earlier works are built on homogeneous networks and fail to capture the notion of heterogeneous signals of node similarities. While PageRank computed the authority scores or influence scores over nodes, the rooted-RW method [72] is a commonly used metric for node similarity computation with respect to a fixed node. In our work in Chapter 6, we extend the authority computation of the PageRank algorithm, and the node similarity computation of the rooted-RW method to a multidimensional relation space. In the future, we aim to extend our work by implementing approximate rooted-RW efficiently using the map-reduce framework in [12], and also study the effect of extending our work to evolving social graphs [13] over forum participants.

The edges in our multidimensional user graph represent similarity between nodes. Several studies have focused on comparing different similarity computations. In [72], the authors compare the effectiveness of about fifteen different similarity measures including the rooted-RW measure for predicting links in a co-authorship network. The studies in [37, 68] find subgraphs that represent the connection between any two nodes in the graph efficiently, and use these subgraphs to compute node proximity. However, these studies do not incorporate multiple user relations. In our work, we define edge weights using cosine similarity or frequency counts of common user behavior, and develop similarity scores across the entire social graph. In the future, we aim to study different edge weight measures and different node-centric similarity measures over our multidimensional user graph.

Recently, the PathSim algorithm [96] was built on heterogeneous graphs and provides node similarity using fixed length pre-defined paths. Another approach in [70] finds answer nodes to a typed query by assigning weights to constrained paths along the random walks. These

pre-defined paths fail to find relations between distant nodes and do not allow for a dynamic selection of relations or paths for similarity computation. In [80], the importance weights of both nodes and relations is computed simultaneously. Our work focuses on finding node similarities in heterogeneous relation graphs, and we find relation importance in an egocentric manner. Unlike the work in [20] where the authors use the content created by participants in explicit online social networks to find expert authors, we do not have explicit social networks and ground truth assessment of author expertise w.r.t. particular queries. We learn these expertise scores from the implicit signals captured from user generated content in forum posts. In addition, we leverage user expertise scores to improve keyword search while the authors in [20] leverage content generated in social networks to find user expertise.

Finding similar users in online data has significant social and economical applications like targeted advertising and marketing, online dating, news dissemination and networking. Predicting links in social networks ([72, 86]) using a variety of similarity measures and user behavior across networks has been studied. Predicting such user behavior is useful in understanding and addressing future user needs. In our work in Chapter 6, we learn the interpersonal relationships amongst online forum participants to predict users who are likely to answer questions posed in a forum thread. Making such accurate predictions can enable users to find information quickly and can also help in making recommendations for building an explicit friendship network.

Chapter 8

Conclusions and Directions for Future Research

We now report on the major conclusions of this thesis in Section 8.1 and propose some future work in Section 8.2.

8.1 Conclusions

In this dissertation we identified and addressed several challenges related to understanding online user preferences and similarities, and using these to enhance automatic applications like search and recommendations. We introduced novel approaches to identify structure over free-form user generated text, developed techniques for building text based recommendation systems, studied the problem of a multi-granularity search system over user data and designed personalization techniques using the inherent social network linking online users.

In Chapter 2, we described challenges in extracting relevant features from text and discovering user sentiment towards these features. We also described some specific challenges in utilizing the free-form user authored text in search and recommendation systems. As demonstrated in later chapters, we address these challenges by allowing users to access text along topical and sentiment dimensions, as well as allowing for varying focus levels over the large amount of data.

In Chapter 3, we presented the user reviews classification and analysis effort performed as part of our URSA project. We show that both topic and sentiment information at the sentence level are useful information to leverage in a review. We developed techniques for manual annotation of labeled data and automatic sentence classification. We then discussed the correlation and differences between the information captured from the rich text authored by online users and the information present in structured metadata. Additionally, we described a highly efficient semi-supervised algorithm for topic discovery that does not require large amount of

human input. Our method approximates the harmonic solution over a graph and we show how highly confident HS predictions on a graph can be identified based on a subgraph. We demonstrated the performance of our method in obtaining nearly optimal semantic labels over words in a graph over user reviews in the restaurant and hotel reviews domain.

We assess the impact of text-derived information in a recommendation system in Chapter 4. We show that both topic and sentiment information at the sentence level are useful information to leverage in a review. In addition, we use soft clustering techniques to group like-minded users for personalized recommendations, using the *detailed textual structure and sentiment of reviews*. Our techniques make better ratings predictions using the textual data, and moreover, we make fine-grained predictions of user sentiment towards individual restaurant features.

In Chapter 5, we presented a novel search system over patient forum data performed as part of the PERSEUS project. Our main contribution is the design of a hierarchical scoring methodology that allows several granularities of forum objects to be scored and compared in a unified fashion. Using our scores, we can generate results that contain a mixed set of objects, dynamically selecting the best level of focus on the data. We designed the efficient OAKS algorithm to generate the optimal-scored non-overlapping result set that ensures no redundant information. We conducted user studies to assess the relevance of the retrieved search results and our experiments clearly show that a mixed collection of result granularities yields better relevance scores than post-only results.

Online users interact with each other due to a variety of reasons ranging from shared topics of interests, similar demographic information like age, gender and location, or same information need at the same time. In Chapter 6, we described a novel multidimensional similarity framework that builds a random walk using heterogeneous relations between users, enabling us to capture user similarity across a variety of reasons in a unified manner. Our heterogeneous framework captures egocentric similarities for a user in our data, and we leverage these similarities to make highly precise predictions on future interactions between users. Finding which users are likely to provide answers to questions posted on a forum improves user search experience in a personalized manner. In addition, we conducted user studies to assess the relevance of search results generated in response to keyword queries. We then enhance keyword search by re-ranking results retrieved by traditional IR scoring by using information on the authority

or users contributing to the forums. Our results demonstrate an improvement in overall search result relevance within as few as top-10 results, as perceived by crowd sourced judges. Thus, we uniformly capture multidimensional similarities between users to enhance search and access over online forums.

8.2 Future Directions

Online reviews and forums constitute an interesting medium for understanding user similarities and preferences. Recommendation systems enable users to find relevant information quickly and easily. In future, we aim to investigate additional refinements to our text-based recommendations, including better text classification strategies and utilizing temporal factors and other available metadata to guide our analysis. We are also interested in unsupervised techniques that dynamically learn the most important features discussed in user reviews, i.e., identifying review components at a more fine-grained level than topics.

In addition, we are interested in the impact of text classification on search over reviews and are implementing tools that allow users to search reviews using topic and sentiment information. An interesting research direction is to use the text classification in spam-detection techniques or to ascribe a quality score to the reviews. Lastly, similar to the study in [29] we are interested in evaluating the performance of our techniques in generating top-k restaurant recommendation lists.

Search over user generated content still requires many refinements. We aim to study additional optimization functions for generating multi-granularity results with different properties of the result set. An interesting direction is to re-rank search results based on changing and evolving user needs. We are currently developing a search interface for representing multi-granularity results in and out of context with visualization tools like highlighting relevant text.

Our hierarchical model to represent multi-granular objects has many real-world applications. Consider for instance, targeted advertising. If one represents ad topics in a hierarchy in a top-down specialization (say having a path like *accessories* $\rightarrow$ *shoes* $\rightarrow$ *women's shoes* $\rightarrow$ *high-heeled shoes*), then our result generation strategy can ensure ideal selection of advertisements to be displayed in the limited real estate available.

Finally, we are investigating additional methods and signals for learning the weights to be associated with the user relations. We are also studying the query-topic specific expertise scores of users as an alternate mechanism to re-rank the keyword search results. We aim to extend our multi-dimensional user similarity work by implementing approximate rooted-RW efficiently using the map-reduce framework in [12], and also study the effect of extending our work to evolving social graphs [13] over forum participants. In the future, we would like to study our techniques of capturing multiple interpersonal relations on datasets containing explicit symmetric or asymmetric friendship relations, in conjunction with the implicit user similarities.

References

[1] Amazon mechanical turk. https://www.mturk.com/.

[2] Lda topic modelling. www.nlp.stanford.edu/software/tmt.

[3] Opennlp. http://opennlp.sourceforge.net/.

[4] Power reviews. http://www.powerreviews.com/social-shopping/news/press_breed_11122007.html, 2007.

[5] Marketing charts. http://www.marketingcharts.com/interactive/, 2008.

[6] L. A. Adamic and E. Adar. How to search a social network. *Social Networks*, 27:2005, 2005.

[7] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94, 2000.

[8] O. Alonso, R. Schenkel, and M. Theobald. Crowdsourcing assessments for xml ranked retrieval. In *ECIR*, pages 602–606, 2010.

[9] S. Amer-Yahia and M. Lalmas. Xml search: languages, inex and scoring. *SIGMOD Rec.*, 35(4):16–23, 2006.

[10] N. Archak, A. Ghose, and P. G. Ipeirotis. Show me the money!: deriving the pricing power of product features by mining consumer reviews. In *SIGKDD*, 2007.

[11] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. Monte carlo methods in pagerank computation: When one iteration is sufficient. *SIAM J. Numer. Anal.*, 45(2):890–904, Feb. 2007.

[12] B. Bahmani, A. Chowdhury, and A. Goel. Fast incremental and personalized pagerank. *Proc. VLDB Endow.*, 4(3):173–184, Dec. 2010.

[13] B. Bahmani, R. Kumar, M. Mahdian, and E. Upfal. Pagerank on an evolving graph. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 24–32, New York, NY, USA, 2012. ACM.

[14] P. Balatsoukas and P. Demian. Effects of granularity of search results on the relevance judgment behavior of engineers: Building systems for retrieval and understanding of context. *JASIST*, pages 453–467, 2010.

[15] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[16] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explorations Newsletter*, pages 75–79, December 2007.

[17] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *IEEE International Conference on Data Mining*, 2007.

[18] M. Bendersky and O. Kurland. Utilizing passage-based language models for document retrieval. In *Proceedings of the IR research, 30th European conference on Advances in information retrieval*, ECIR'08, pages 162–174, 2008.

[19] J. Bennett and S. Lanning. The netflix prize. In *KDD Cup and Workshop*, 2007.

[20] A. Bozzon, M. Brambilla, S. Ceri, M. Silvestri, and G. Vesci. Choosing the right crowd: expert finding in social networks. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 637–648, 2013.

[21] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Conference on Uncertainty of Artificial Intelligence*, pages 43–52, 1998.

[22] S. Brin. Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT 98*, pages 172–183, 1998.

[23] S. Brody and N. Elhadad. An unsupervised aspect-sentiment model for online reviews. In *HLT-NAACL Conference of the North American Chapter of the Association for Computational Linguistics*, 2010.

[24] J. A. Bullinaria and J. P. Levy. Extracting semantic representations from word co-occurrence statistics: A computational study.

[25] D. Carmel, N. Zwerdling, I. Guy, S. Ofek-Koifman, N. Har'el, I. Ronen, E. Uziel, S. Yogev, and S. Chernov. Personalized social search based on the user's social network. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1227–1236, New York, NY, USA, 2009. ACM.

[26] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16:22–29, 1990.

[27] C. L. A. Clarke. Controlling overlap in content-oriented xml retrieval. In *SIGIR*, pages 314–321, 2005.

[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. 2001.

[29] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46, 2010.

[30] B. W. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*. 2009.

[31] B. T. Dai, N. Koudas, B. C. Ooi, D. Srivastava, and S. Venkatasubramanian. Rapid identification of column heterogeneity. *IEEE International Conference on Data Mining*, pages 159–170, 2006.

[32] K. Dave. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *WWW*, pages 519–528, 2003.

[33] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28, 1979.

[34] H. Duan and C. Zhai. Exploiting thread structures to improve smoothing of language models for forum post retrieval. In *ECIR*, pages 350–361, 2011.

[35] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in knowitall: (preliminary results). In *WWW*, pages 100–110, 2004.

[36] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. In *Proceedings of ACM-SIAM symposium on Discrete algorithms*, pages 28–36, 2003.

[37] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 118–127, New York, NY, USA, 2004. ACM.

[38] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *Advances in Neural Information Processing Systems 22*, pages 522–530, 2009.

[39] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 801–810, 2005.

[40] M. Futschik and B. Carlisle. Noise-robust soft clustering of gene expression time-course data. In *Journal of Bioinformatics and Computational Biology*, 2005.

[41] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thie. Learning attribute-to-feature mappings for cold-start recommendations. ICDM '10, pages 176–185, 2010.

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

[43] G. Ganu, Y. Kakodkar, and A. Marian. Improving the quality of predictions using textual information in online user reviews. In *Information Systems*, 2013.

[44] G. Ganu and B. Kveton. *Nearly Optimal Semi-Supervised Learning on Subgraphs*, 2013. DCS Technical Report No. 705.

[45] G. Ganu and A. Marian. One size does not fit all: Multi-granularity search of web forums. In *Proceedings of the 22nd international conference on Information and knowledge management*, CIKM '13, 2013.

[46] G. Ganu, A. Marian, and N. Elhadad. *URSA - User Review Structure Analysis: Understanding Online Reviewing Trends*, 2010. DCS Technical Report No. 668.

[47] M. D. Garris, J. L. Blue, G. T. Candela, G. T. C, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C. L. Wilson. Nist form-based handprint recognition system. 1994.

[48] J. Goldberger, H. Greenspan, and S. Gordon. Unsupervised image clustering using the information bottleneck method. In *DAGM Symposium on Pattern Recognition*, 2002.

[49] R. González-Ibáñez, S. Muresan, and N. Wacholder. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, pages 581–586, 2011.

[50] T. H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 517–526, New York, NY, USA, 2002. ACM.

[51] M. A. Hearst. Clustering versus faceted categories for information exploration. *Communications of the ACM*, pages 59–61, 2006.

[52] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, pages 5–53, 2004.

[53] B. Hjørland. The foundation of the concept of relevance. *JASIST*, pages 217–237, 2010.

[54] M. Hu and B. Liu. Mining and summarizing customer reviews. In *SIGKDD*, pages 168–177, 2004.

[55] M. Jahrer, A. Töscher, and R. Legenstein. Combining predictions for accurate recommender systems. In *SIGKDD*, pages 693–702, 2010.

[56] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 271–279, New York, NY, USA, 2003. ACM.

[57] M. Jha and N. Elhadad. Cancer stage prediction based on patient online discourse. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, pages 64–71, 2010.

[58] T. Joachims. A support vector method for multivariate performance measures. In *International Conference on Machine Learning*, 2005.

[59] D. S. Johnson and C. H. Papadimitriou. On generating all maximal independent sets. *Inf. Process. Lett.*, pages 119–123, 1988.

[60] H. Joho and J. M. Jose. Effectiveness of additional representations for the search result presentation on the web. *Information Processing and Management*, pages 226–241, 2008.

[61] T. Kanungo, N. Ghamrawi, K. Y. Kim, and L. Wai. Web search result summarization: title selection algorithms and user satisfaction. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1581–1584, 2009.

[62] T. Kanungo and D. Orr. Predicting the readability of short web summaries. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 202–211, 2009.

[63] J. Kekäläinen and K. Järvelin. Using graded relevance assessments in ir evaluation. *JASIST*, pages 1120–1129, 2002.

[64] C. W. ki Leung, S. C. fai Chan, and F. lai Chung. Integrating collaborative filtering and sentiment analysis: A rating inference approach. In *ECAI-Workshop on Recommender Systems*, pages 62–66, 2006.

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

[66] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*, pages 426–434, 2008.

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

[68] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity graphs in networks. *ACM Trans. Knowl. Discov. Data*, 1(3), Dec. 2007.

[69] B. Kveton, M. Valko, A. Rahimi, and L. Huang. Semi-supervised learning with max-margin graph cuts. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 421–428, 2010.

[70] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Mach. Learn.*, 81(1):53–67, 2010.

[71] G. Lever, T. Diethe, and J. Shawe-Taylor. Data dependent kernels in nearly-linear time. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, pages 685–693, 2012.

[72] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.

[73] K.-I. Lin and R. Kondadadi. A similarity-based soft clustering algorithm for documents. In *DASFAA: International Conference on Database Systems for Advanced Applications*, pages 40–47, 2001.

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

[75] B. Liu, M. Hu, and J. Cheng. Opinion observer: Analyzing and comparing opinions on the web. In *WWW*, pages 342–351, 2005.

[76] Y. Lu, C. Zhai, and N. Sundaresan. Rated aspect summarization of short comments. In *WWW*, 2009.

[77] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. 2008.

[78] W. Mendenhall and T. L. Sincich. *A Second Course in Statistics: Regression Analysis (6th Edition)*. Prentice Hall.

[79] V. Mihajlović, G. Ramírez, T. Westerveld, D. Hiemstra, H. E. Blok, and A. P. de Vries. Tijah scratches inex 2005 vague element selection, overlap, image search, relevance feedback, and users. In *INEX Workshop*, pages 54–71, 2005.

[80] M. K.-P. Ng, X. Li, and Y. Ye. Multirank: co-ranking for objects and relations in multi-relational data. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 1217–1225, New York, NY, USA, 2011. ACM.

[81] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

[82] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas. Web-scale distributional similarity and entity set expansion. In *EMNLP*, pages 938–947, 2009.

[83] P. Pantel and M. Pennacchiotti. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *ACL*, pages 113–120, 2006.

[84] J. Pehcevski. Relevance in xml retrieval: The user perspective. In *SIGIR*, 2006.

[85] N. Pharo. The effect of granularity and order in xml element retrieval. *Information Processing and Management*, pages 1732–1740, 2008.

[86] G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Link prediction across networks by biased cross-network sampling. In *ICDE*, pages 793–804, 2013.

[87] G. Ramírez and A. P. de Vries. Relevant contextual features in xml retrieval. In *Proceedings of the 1st international conference on Information interaction in context*, pages 56–65, 2006.

[88] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[89] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, pages 59–66, 1988.

[90] K. Sauvagnat, L. Hlaoua, and M. Boughanem. Xfirm at inex 2005: ad-hoc and relevance feedback track. In *INEX*, pages 88–103, 2005.

[91] S. Siegel and J. N. John Castellan. *Nonparametric Statistics for the Behavioral Sciences, Second Edition*. McGraw-Hill, 1988.

[92] N. Slonim. *The Information Bottleneck: Theory and Applications*. PhD thesis, Hebrew University, 2002.

[93] N. Slonim, N. Friedman, and N. Tishby. Unsupervised document classification using sequential information maximization. In *SIGIR*, 2002.

[94] P. Sondhi, M. Gupta, C. Zhai, and J. Hockenmaier. Shallow information extraction from medical forum data. In *COLING*, pages 1158–1166, 2010.

[95] D. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 81–90, 2004.

[96] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proc. VLDB Endow.*, 4(11):992–1003, 2011.

[97] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Matrix factorization and neighbor based algorithms for the netflix prize problem. In *RecSys*, pages 267–274, 2008.

[98] A. Talwalkar, S. Kumar, and H. Rowley. Large-scale manifold learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[99] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Annual Allerton Conference on Communication, Control and Computing*, 1999.

[100] I. Titov and R. McDonald. A joint model of text and aspect ratings for sentiment summarization. In *ACL Annual Meeting of the Association of Computational Linguistics*, 2008.

[101] P. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL, 2001.

[102] M. Valko, B. Kveton, L. Huang, and D. Ting. Online semi-supervised learning on quantized graphs. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 2010.

[103] V. V. Vazirani. *Approximation Algorithms*. 2004.

[104] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *SIGKDD*, pages 448–456, 2011.

[105] G. Xu and W.-Y. Ma. Building implicit links from content for forum search. In *SIGIR*, pages 300–307, 2006.

[106] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[107] Z. yu Niu and D. hong Ji. Word sense disambiguation using label propagation based semi-supervised learning. In *ACL*, 2005.

[108] W. Zheng, H. Fang, and C. Yao. Exploiting concept hierarchy for result diversification. In *CIKM*, pages 1844–1848, 2012.

[109] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, University of Wisconsin-Madison, 2008.

[110] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912–919, 2003.