Discriminative Clustering Algorithms for Document Understanding, Tag Recommendation, and Web Surfer Behavior Prediction

by

Malik Tahir Hassan

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Syed Babar Ali School of Science and Engineering at the Lahore University of Management Sciences, Lahore, Pakistan

2013

Major Professor

Dr. Asim Karim
Dedicated to

the pride of this universe

MUHAMMAD (Peace Be Upon Him)

and

the source of my strength

my family
Acknowledgements

In the name of Allah the most beneficent the most merciful. All praise be to Him who has blessed us with kind people around us.

I express my deepest gratitude to my supervisor Dr. Asim Karim. A meeting with him is always a confidence booster.

I acknowledge the support of my family specially my mother and my wife for their prayers and patience. Also, I acknowledge the help of my PhD fellows Junejo, Fahad, Saqib, Rana, Osama, Jamal and all rest readily available through out the course of this work.

This research was funded by Higher Education Commission, Islamabad, Pakistan and Lahore University of Management Sciences, Lahore, Pakistan. Their support is gratefully acknowledged.
Abstract

The Web is a goldmine of knowledge, but its realization requires effective and efficient discovery algorithms. Information on the Web ranges from textual documents to social content to usage patterns. Such information is huge and dynamic in nature making useful knowledge discovery a challenging task. In recent years, data mining techniques have been utilized for various knowledge discovery tasks with success. Data clustering, in particular, has two key advantages for Web mining: (1) it is an unsupervised technique that does not require labeled data; (2) it is a conceptually simple task that can produce readily understandable patterns. In this thesis, we develop and evaluate discriminative clustering algorithms for textual document understanding, social content tag recommendation, and Web surfing behavior analysis. Our discriminative clustering algorithms are efficient and semantically rich for effective knowledge discovery on the Web.

For textual document clustering and understanding, we develop and evaluate a new algorithm called CDIM (Clustering via Discrimination Information Maximization). CDIM is an iterative partitional clustering algorithm that maximizes the sum of discrimination information provided by documents in the collection. A key advantage of CDIM is that its clusters are describable by their highly discriminating terms, or equivalently, their highly topically-related terms. This is achieved by incorporating statistically sound measures of discrimination that have been shown to convey semantic relatedness of terms to topics into the clustering algorithm. A hierarchical version of CDIM is also presented. CDIM’s superior performance is demonstrated on benchmark datasets in comparison with current state-of-the-art text clustering algorithms.

For social content tag recommendation, we develop a model of contents and tags using CDIM for recommendation of tags of new content. User textual posts (contents) are clustered to yield a list of discriminative terms for each cluster. Likewise, textual tagging history is clustered to produce another list of terms. These lists are combined with user’s personal tagging history, if available, to produce the final tag recommendations. Our approach is evaluated on the data of a social bookmarking system Bibsonomy. We observe that the recommendation accuracy can be improved by
updating the recommendation model from time to time. To realize this in an efficient manner, we build a self-optimizing version of our tag recommendation system. The self-optimization strategy decides when and how to update the system by solving a nonlinear optimization problem constrained on available time to decide the best clustering parameters (number of clusterable records and number of clusters). A better alternate to re-building the complete clustering models is doing corrections to clusters that are getting outdated and are contributing to errors. We achieve this by developing a self-calibration strategy for our system which is shown to be a better and more practical option. We also perform an analysis of personalized and non-personalized versions of our tag recommendation system. Besides our discriminative clustering based tag recommendations algorithm, performance of other algorithms including PITF (Pair wise Interaction Tensor Factorization), FolkRank, and adapted PageRank is analyzed on our proposed personalization groups (beginners, followers, and leaders) in folksonomies.

For Web surfer behavior analysis, we find patterns of Web navigation paths among users and then develop discriminative and generative models for predicting future paths of users. Navigation patterns or behaviors are discovered by adapting the k-modes clustering algorithm with a new similarity measure appropriate for comparing navigation paths and a new method for cluster initialization. Our experiments, conducted on two real-world datasets, demonstrate that predictions based on navigation behaviors are not necessarily better because of diversity of behaviors on the Web. Likewise, it is found that inclusion of start time of navigation sessions in predication models has little affect on accuracy but is significantly bad on efficiency. On the other hand, predictions based on cluster centroids are very cost-efficient without significant loss in accuracy.

This thesis demonstrates the usefulness and versatility of clustering algorithms for Web mining, and highlights the importance of semantics in textual document analysis and self-management in practical Web systems. Directions for future work include semantic enhancements to CDIM and developments of self-management strategies for data mining applications.
List of Author Publications


9. “CDIM: Clustering and Understanding Documents by Discrimination Information Maximization”, Malik Tahir Hassan, Asim Karim; IEEE Transactions on Knowledge and Data Engineering (TKDE), manuscript under review, 2013.
## Contents

1 Introduction and Motivation .................................................. 1

1.1 Web Mining and Clustering ................................................. 1

1.2 Applications of Clustering and Web Mining ............................. 3

1.2.1 Topic Identification, Document Understanding and Summarization 4

1.2.2 Digital Book Clustering .............................................. 4

1.2.3 Clustering and Web Search .......................................... 4

1.2.4 Clustering and Web Advertising .................................... 5

1.2.5 Clustering in Social Web Mining .................................... 5

1.3 Challenges in Clustering Based Web Mining ............................. 7

1.3.1 The Web is Enormous ............................................... 8

1.3.2 Unstructured and Unlabeled Data .................................. 8

1.3.3 High Dimensional and Sparse Data ................................. 8

1.3.4 Ambiguous, Uncertain and Missing Data ............................ 9

1.3.5 Multilingual and Informal Data .................................... 9

1.4 Our Contribution .......................................................... 9

1.4.1 Major Contributions ................................................ 11

1.4.2 Minor Contributions ................................................ 12

1.4.3 Related Publications ............................................... 13

1.5 Thesis Outline ................................................................ 14
## 2 Background and Related Work

2.1 Mining the World Wide Web ................................. 16
2.1.1 Evolution of the Web ........................................ 17
2.1.2 Web Mining .................................................... 18
2.1.3 Types of Web Mining ........................................ 18
2.1.4 Decade of Web Mining ....................................... 19
2.1.5 Tag Recommendation Systems .............................. 20
2.1.6 Self-Optimization and Self-Calibration ..................... 23
2.1.7 Web Surfing Behavior Prediction and Clustering ........... 24

2.2 Clustering ....................................................... 26
2.2.1 Clustering Methods ......................................... 26
2.2.2 Document Clustering ........................................ 28
2.2.3 Discrimination Information .................................. 31

2.3 Conclusion ....................................................... 31

## 3 Discriminative Clustering for Text

3.1 Introduction .................................................... 32
3.2 CDIM – Our Document Clustering Method ...................... 34
3.2.1 Problem Statement .......................................... 34
3.2.2 Clustering Objective Function ............................. 35
3.2.3 Term Discrimination Information .......................... 36
3.2.4 Smoothing ................................................... 38
3.2.5 Relatedness of Terms to Clusters .......................... 38
3.2.6 Document Discrimination Information ...................... 39
3.2.7 Algorithm ................................................... 40
3.2.8 Key Characteristics of CDIM .............................. 41
3.2.9 Convergence of CDIM ...................................... 42

3.3 Variations of CDIM ............................................. 43
3.3.1 CDIM-RR and CDIM-MDI ................................. 43
3.3.2 CDIM by Repeated Bisection ........................................ 44
3.3.3 CDIM with Information Retrieval Measures DR and DC .......... 44

3.4 Experimental Setup ..................................................... 45
3.4.1 Data Sets .......................................................... 46
3.4.2 Comparison Methods .............................................. 47
3.4.3 Clustering Validation Measures .................................. 48

3.5 Results and Discussion ................................................ 49
3.5.1 Clustering Quality ............................................... 49
3.5.2 Cluster Understanding and Visualization ......................... 52
3.5.3 Additional Experiments and Discussion ......................... 53

3.6 Conclusion ........................................................... 56

4 Content-Based Tag Recommendation using Discriminative Clustering 57
4.1 Introduction ............................................................ 57
4.2 Discriminative Clustering for Content Based Tag Recommendation .... 58
4.2.1 Problem Definition and Notation ................................ 59
4.2.2 Discriminative Clustering for Tag and Term Ranking .......... 60
4.2.3 Final Tag Recommendation ...................................... 62

4.3 Evaluation Setup ...................................................... 63
4.3.1 Data and their Characteristics .................................. 63
4.3.2 Data Preparation .................................................. 65
4.3.3 Evaluation Criteria ............................................... 66

4.4 Results ............................................................... 66
4.4.1 Clustering Performance ......................................... 66
4.4.2 Tag Recommendation Using $TG$ and $TM$ Only ............... 67
4.4.3 Tag Recommendation Using All Lists .......................... 68
4.4.4 Tag Recommendation for Task 1 Test Data of ECML DC 2009 ... 70

4.5 Conclusion ............................................................ 70

xi
6.3 Clustering-Prediction Framework for Web Surfer Behavior Prediction .......................... 102
  6.3.1 Behavior Clustering ........................................... 103
  6.3.2 Behavior Prediction ........................................... 108

6.4 Experimental Evaluation ........................................... 112
  6.4.1 Data and their Characteristics ................................ 112
  6.4.2 Evaluation Criteria ........................................... 114
  6.4.3 Results and Discussion ....................................... 116

6.5 Conclusion ......................................................... 121

7 Conclusion and Future Work ........................................... 123
  7.1 Thesis Conclusion ................................................ 123
  7.2 Future Directions ............................................... 126
    7.2.1 CDIM Extensions ........................................... 126
    7.2.2 Tag Recommendation ....................................... 127
    7.2.3 Web surfer behavior prediction ............................. 128
List of Figures

1.1 Example of clustering. Unlabeled documents are separated into three groups (may be Technology, Sports and Health) based on their terms. ............................................. 2
1.2 Branches of Web mining and scope of this thesis, items in italic are out of scope ........ 3
3.1 Terms per document value can be used to select between RR or MDI ........................... 51
3.2 Scatter plot of documents projected onto the 2-D discrimination information space. True labels are indicated by different color markers. ........................................... 54
3.3 Impact of term selection on clustering quality ......................................................... 55
3.4 Convergence curves of CDIM ................................................................................. 55
4.1 Number of tags assigned to posts by users ............................................................ 64
4.2 Discriminative clustering convergence curves (clustering posts based on tags) ........... 68
5.1 Clustering time versus both $N$ and $K$ .................................................................... 78
5.2 Fscore versus both $N$ and $K$ ................................................................................ 79
5.3 Performance variation with number of test posts (best performance at about 6,000) ... 80
5.4 Comparison between original and re-clustering based recommendation systems ...... 81
5.5 Performance of original, re-clustering, and self-optimizing tag recommendation systems 82
5.6 Comparison of cluster accuracy with and without self-calibration .......................... 85
5.7 Distribution of users posts in Flickr .......................................................................... 88
5.8 Life cycle of a Folksonomy User ............................................................................... 90
5.9 Unique Tag Gain in Life Cycle of a Folksonomy ..................................................... 92
5.10 Prediction Performance of PITF on Bibsonomy Dataset .......................................... 94
6.1 Overview of Clustering-Prediction Framework ........................................ 103
6.2 Average intra-cluster similarity for different values of $k$ (ECML data) ........ 107
6.3 Average intra-cluster similarity for different values of $k$ (KDD data) ........ 107
6.4 Distribution of users across behavior clusters for $k = 10$ and $k = 19$ (ECML data) . 115
6.5 Distribution of users across behavior clusters for $k = 20$ and $k = 45$ (KDD data) . 115
List of Tables

3.1 Data sets and their characteristics ........................................ 47
3.2 Data sets statistics (rounded off values). Terms (M), Documents (N), Categories (K). 50
3.3 Comparison of clustering quality of CDIM-RR via the BCubed F-measure. Averages and standard deviations obtained from 10 runs are shown. .......................... 51
3.4 Comparison of clustering quality of CDIM-MDI via the BCubed F-measure. Averages and standard deviations obtained from 10 runs are shown. .......................... 51
3.5 Comparison of clustering quality among CDIM variants via the BCubed F-measure. Averages and standard deviations obtained from 10 runs are shown. .......................... 52
3.6 Top 10 most discriminating terms (stemmed words) for clusters in ohscal data set .......................... 53
4.1 Post-core at level 2 data statistics ........................................ 64
4.2 Performance of discriminative clustering of posts using the tags assigned to them (post-core at level 2 data) ........................................ 67
4.3 Top tags for selected clusters (K = 200) ........................................ 68
4.4 Tag recommendation performance (average F1-score percentages) using $T_G$ or $T_M$ only for original data ........................................ 69
4.5 Tag recommendation performance (average F1-score percentages) for processed data (K = 200; prediction based on top 3 clusters). The bottom line shows performance on ECML-DC09 task 1 test data ........................................ 69
5.1 Statistics of the Three Social Media Datasets ........................................ 87
5.2 Reuse of Items in the Three Social Network Datasets ........................................ 88
5.3 users’ tagging behavior in Flickr ........................................ 89
5.4 Correlation between UTG and UHS ................................. 90
5.5 Importance of UTS in High History Users ........................... 90
5.6 F1 Score for top 5 tag recommendations by graph based algorithms on three subdi- vided datasets ................................................................. 93
5.7 F1 Score for top 5 tag recommendations using content based algorithm, Tagging by Discriminative Clustering, for Bibsonomy data ...................................................... 95
6.1 MatchScore calculation example ....................................... 105
6.2 Bayesian prediction percent scores (ECML data) ......................... 118
6.3 Bayesian prediction percent scores (KDD data) .......................... 118
6.4 SVM prediction percent accuracies and percent scores (ECML data) .................. 118
6.5 SVM prediction percent accuracies and percent scores (KDD data) .................. 118
6.6 Bayesian prediction percent scores with and without timestamp information (without clustering) ........................................ 120
6.7 SVM prediction percent accuracies and percent scores with and without timestamp information (without clustering) ........................................ 120
6.8 Prediction percent scores obtained when using most relevant cluster mode ............. 121
Chapter 1

Introduction and Motivation

1.1 Web Mining and Clustering

The Web is a success because of Web mining. Think of the Web where there is no Google, no YouTube, no Facebook; it becomes a useless resource, a boring place, a quagmire. It is Web mining that has played an undeniable role in success of these glorious, engaging and powerful applications, heightening the value of the Web overall.

Web mining has been under keen focus of data mining community for more than a decade [Spiliopoulou et al. (2012)]. Applying data mining techniques e.g. clustering, to discover knowledge from the Web is called Web mining.

Making groups of similar objects is called clustering (Figure 1.1). In the context of the Web, these objects can be Web users, Web pages and Web page contents. Clustering is a very fundamental data mining and machine learning technique that has been applied to discover very valuable knowledge from the Web. It belongs to the set of unsupervised learning methods as it does not require labeled data for training. In literature clustering is also referred as segmentation, partitioning and unsupervised classification [Tan et al. (2006)].

Selection of clustering based methods to mine the Web becomes a natural choice due to the availability of huge amounts of unlabeled but valuable data present on or through the Web e.g. Web pages, articles, news, emails, blogs, tweets, user profiles, navigation histories etc. The knowledge
gained through clustering the Web data can lead to develop applications that are more accurate, more efficient and more adaptive as compared to the solutions that do not use this precious resource [Miner et al. (2012)].

Figure 1.1: Example of clustering. Unlabeled documents are separated into three groups (may be Technology, Sports and Health) based on their terms.

Web mining is generally divided into three branches; Web Content Mining, Web Structure Mining and Web Usage Mining [Kosala and Blockeel (2000)]. While the other two branches namely Web Content Mining and Web Structure Mining are more concerned about the fabric and makeup of Web pages, Web Usage Mining discovers patterns in the usage of these Web pages. Web Content Mining discovers knowledge from the contents of the Web pages. Web structure mining aims to find useful knowledge from the structure of the Web pages i.e. hyperlinks.

This thesis presents novel text clustering methods and targets the content and usage types of Web mining where content means the text content specifically. Figure 1.2 summarizes the branches of Web mining under focus.
1.2 Applications of Clustering and Web Mining

Regardless of the size of the organization, getting online by providing a functional and up to date Web site has become essential now a days and proper use of Web mining guarantees competitive advantage. Even individuals who maintain a Web page, blog or a social network profile are considered more knowledgeable and well-connected. Friend prediction, query completion, personalized resource recommendation e.g. article, music or video etc. are some of the Web mining systems that facilitate browsing activities of individual users by applying clustering methods. Predicting potential customers, faster and targeted marketing, outlier and fraud detection, document summarization, sentiment analysis, quick information search and retrieval are some other examples where Web mining via clustering plays a vital role for individuals as well as organizations.

As the Web flourishes, huge amounts of Web usage and Web content data are available for application of data mining techniques. We identified above many successful Web applications that learn interesting patterns through clustering of the available unlabeled data and build better new applications or improve existing ones. Some of the exemplary application areas are highlighted below that provide motivation to develop novel clustering algorithms for Web mining.
1.2.1 Topic Identification, Document Understanding and Summarization

Text document clustering has been applied popularly to the three important research problems. Information on the Web in the form of news or research articles, blogs, tweets etc. need discriminating topics for better organization and presentation purposes. Note that, even labeled documents need selection of representative terms to be shown as document topic [Cheng et al. (2012), Shen et al. (2011), Aliguliyev (2009)]. A good text clustering algorithm that can pick relevant terms from clusters and documents can be helpful in solving the topic identification problem. A document clustering method e.g. CDIM [Hassan and Karim (2012)] can also provide a ranking of discriminating terms for clusters as well as documents that can help in understanding a document plus preparing a summary of this document.

1.2.2 Digital Book Clustering

Digital libraries have lessened the traffic to traditional libraries. Easier and cheaper availability of electronic documents and books has originated the interesting clustering and Web mining application, digital book clustering. Modified as well as new text document clustering algorithms step forward to cluster digital books [Spasojevic and Poncin (2011), Lidong and Yuan (2012)]. The application is not trivial as the size of a digital book is normally very large.

1.2.3 Clustering and Web Search

Web search is one of the most popular online activities and a key method to retrieve desired information. World Wide Web without good search engines like Google, Bing, Yahoo, CLUSTY etc. seems worthless. You give some relevant keywords (search query) and your required information is just a click away (most of the times).

Search result clustering (SRC) is a famous application where results from one or more search engines are grouped together into clusters that are coherent topically. Identification of meaningful and relevant phrases as topic labels from the cluster members is also a part of SRC. Clustering can also help in query refinement by improving the selection and placement of query suggestions using document click and session information [Lee et al. (2009), Scaiella et al. (2012)].
Clustering is also useful in developing storylines from high ranking search results and displaying them to the user instead of showing results in the traditional way [Mahadevan et al. (2011)].

1.2.4 Clustering and Web Advertising

Web advertising focuses on the triangle made up of ad, user and context. Behavioral clustering combined with classification strategies can help in deciding the optimal third corner, given the other two. Decide best ad for a user in a context, decide best user(s) for an ad in a context, and decide best context for an ad for a user. In contrast to traditional advertising, Web advertising has generally larger audience, larger number of ads and quantifiable personalization possibilities. Clustering also helps in targeted Web advertising by segmenting the consumers in target groups [Mitra and Baid (2009), Farahat and Bailey (2012)] .

1.2.5 Clustering in Social Web Mining

The Web that enables social interactions of its users with each other is called social Web. Social networking websites, online shopping, education, gaming etc. are all part of the social Web. Sites like Facebook, YouTube and Twitter also provide connectivity to their users via mobile platforms. Wikis and blogs are also key examples of social Web. Wikis can be constructed and updated by common people (access required). In a blog normally an individual shares some information on which others can comment and respond. Product reviews is another important social aspect of Web sites e.g. Amazon. Comments and rating section on such a Web site is one of its most attractive and widely browsed part. Social Web also facilitates collaborative efforts like development of open-source software projects e.g. Linux and Android.

Mining the social Web can yield practically useful knowledge. Clustering and classification of micro-blogging data, for example, can help in discovering emerging topics in social streams. Text clustering is used in Wikification that can help in producing very comprehensive online documents. Clustering and summarization of product reviews to generate recommendations for users is another application with high business value. Social networks can be analyzed to hear people’s voices on different controversies and to estimate popularity ranking e.g. before elections. In essence, social
Web can be considered a red-hot venue for Web Mining through text clustering.

**Recommenders**

With the advent of online social networks and folksonomies, another type of systems got attention of the research community. The recommender systems that suggests resources e.g. news, music, video or labels (also called tags or keywords) to the online resources. Friends and community recommendations by Facebook, video recommendations by YouTube, music recommendations by last.fm are all predictions by sophisticated recommendation systems that use a mix of collaborative filtering, content filtering and personalization techniques [Wei *et al.* (2011)].

In most of the Web 2.0 applications and online social networks users share resources and assign keywords to these resources. These keywords are commonly known as tags. Tagging helps in better organization, sharing, search and retrieval of information. Automatic tag recommenders suggest relevant tags to users when they share resources. Web mining offers many graph based, content based, personalized and non-personalized algorithms to recommend relevant tags with good accuracy. Clustering can help the recommenders in preparing target groups that can be suggested similar/clustered items [Shepitsen *et al.* (2008), Leginus *et al.* (2012)].

**Sentiment Analysis or Opinion Mining**

Sentiment analysis or opinion mining analyzes people’s opinions, emotions, sentiments and reviews from their writings. For example, judging the polarity of a product review to be positive, negative or neutral is an opinion mining task. Different statistics, machine learning and natural language processing techniques have been applied in this area. The importance of sentiment analysis has raised with the rise of Web 2.0. More individuals are participating in discussions, reviews, blogs, ratings etc. producing data that hides very valuable knowledge. Opinion mining is a challenging Web mining task but has huge practical as well as social value. An in time mining of sentiments and feedback from users can help businesses correct their errors. Even states can learn and make right decisions by mining the sentiments of their people regularly [OConnor *et al.* (2010), Hu *et al.* (2013)]. The unsupervised nature of this domain also welcomes the application of effective and
efficient clustering methods.

**Web Security**

With exponentially increasing information and user population on the Web, securing the information and users has become imperative. Especially, social Web has increased the number of connections hugely which exposes both the users and the Web pages to a number of threats including privacy attacks, viruses, online frauds, spam etc. For instance, spam has grown irritatingly common that you find it in emails, blogs, Web searches, social networks and product reviews etc. Web mining based on clustering and classification algorithms can filter spam, detect intrusions, and predict fraudulent behaviors quite effectively [Phua *et al.* (2010), Christou *et al.* (2011)].

**Mining the Crowd’s Data**

Outsourcing of jobs to large groups of people instead of your company employees or contractors is called crowdsourcing. It comes with a reasonable assumption that a group of people is often more intelligent than an individual. Although crowdsourcing can be offline e.g. volunteers were used to index English terms and example quotes for Oxford English Dictionary, online crowdsourcing is much easier, cheaper and faster. We see many examples of crowdsourcing that use Wisdom of the Crowd very successfully. Wikipedia is an example. Multinational companies involve people in designing their products through crowdvoting. News sites also take online polls and votes to judge popularity. Crowdfunding is another type of crowdsourcing that is used to raise funds for different projects. Automatic opinion mining, clustering and classification of participants and participation, trend prediction, document summarization etc. are the Web mining activities that can be directly applied to the data obtained through crowdsourcing [Barbier *et al.* (2012), Jiang (2013)].

### 1.3 Challenges in Clustering Based Web Mining

After establishing the need for clustering based Web mining algorithms with motivating examples and applications, this section highlights key challenges involved in developing such algorithms for knowledge discovery.
1.3.1 The Web is Enormous

The size of the Web is enormous, and is still growing. Google\textsuperscript{1} reports more than one trillion (1,000,000,000,000) unique URLs on the Web. The Indexed Web contains at least 14.04 billion pages (Sunday, 20 January, 2013)\textsuperscript{2}. In connection, there are billions of users that are interacting with the Web looking for interesting information. This huge size of the Web requires mining algorithms that are very efficient and scalable.

1.3.2 Unstructured and Unlabeled Data

Although data on the Web is not fearfully random, it is not welcoming structured too. Web pages may contain data with annotations and in structures like html or xml tags but generally the data is considered unstructured or semi-structured. Efforts to harness collective intelligence in Web 2.0 applications elevates this issue further. Mining this semi-structured type of data on the Web is pretty challenging and solutions must be accurate as well as robust.

Getting labeled data is usually expensive in time wise as well money. Most of the data on the Web is unlabeled available in the forms of emails, news, research articles, blogs, reviews etc. Mining interesting knowledge from this type of data is not easy. Web mining researchers use interesting unsupervised methods as well as semi-supervised ways to utilize abundantly available unlabeled data with lesser amount of labeled data.

1.3.3 High Dimensional and Sparse Data

Web mining algorithms have to cope with the challenges of very high dimensional and sparse data. Data present on the Web easily contains hundreds and thousands of features. Take an example of text documents e.g. emails, sms, tweets etc. Every word in the dataset (or dictionary) is taken as a feature normally and the size of the feature set reaches thousands. In addition, most of the documents contain a small number of words in all causing high data sparsity. Thus intelligent mining techniques are required to develop space and time efficient knowledge discovery systems.

\textsuperscript{1}http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html, July 2008
\textsuperscript{2}http://www.worldwidewebsize.com/Sunday, 20 January, 2013
1.3.4 Ambiguous, Uncertain and Missing Data

Another challenge in Web mining is handling of ambiguous, uncertain and missing data. Ambiguous data means data that can have more than one meanings. For example the word ‘bank’ can be a financial institution or edge of a river. Understanding the context helps in resolving data ambiguity and is not an easy task. Missing data is another well-known challenge for Web mining community. Neither users nor systems are in habit of giving complete data readily. Data is called uncertain when it is not predictable or precise. Common causes of uncertain data are outdated information sources, imprecise measurements and inexact inputs etc. For example, most of the Web searches usually start with imprecise query terms as users may not know how to convey their requirements precisely. Similarly some data sources contain items/objects with existential probabilities e.g. transactional records, satellite images, medical data etc. instead of discrete values. Positive result of test ‘x’ implies 60% probability of suffering from disease ‘y’ is example of such a record.

1.3.5 Multilingual and Informal Data

Language of expression on the Web is not limited by any hard standards. Although English is dominant, but you can find Web pages in all major languages. Wherever possible, users from different origins are happy to participate in their native languages. People maintain blogs, write comments and reviews, give feedbacks and communicate with each other in different languages even in scripts that use alphabets of one language to form words of some other language e.g. ‘acha kaam’ sentence used by an Urdu native means ‘good work’. In addition, informal language where words are intentionally misspelled or shortened e.g. ‘i m f9, h r u?’ meaning ‘i am fine, how are you?’ and use of slang like ‘dejuno’ for ‘did you know?’ are pretty common. Extracting knowledge from this amalgamated Web is a challenging task.

1.4 Our Contribution

Data mining is a fertile research domain with demonstrated applications in all the fields where useful knowledge is to be discovered from huge data repositories like the Web. Novel algorithms,
effective methods and efficient techniques are popping up the screen all the time. This thesis
is another advancement in state of the art of Data Mining. We present and discuss two novel
clustering algorithms CDIM (Clustering via Discrimination Information Maximization) and Seq-
Clust (Sequence Clustering) with examples of their successful applications in Web mining.

The first algorithm, CDIM (Clustering via Discrimination Information Maximization), is a
document clustering algorithm capable of producing high quality clusters plus a ranking of dis-
criminating terms for each of these clusters. The computational time of CDIM depends linearly on
the clustering parameters. CDIM can handle sparse, high dimensional, multilingual, unstructured
and unlabeled data successfully (details in chapter 3).

CDIM is thoroughly tested on datasets of different contexts and sizes. Results prove its better
performance as compared to other widely used text clustering algorithms including different vari-
ants of \(k\)-means and NMF. Besides content mining from the traditional Web or Web 1.0, CDIM
is also tested on Mining the content of Web 2.0. A tag recommendation system is built for a
social bookmarking system using CDIM. The historical tag assignment data and descriptive terms
about posts (bookmarks and BibTeX items) are clustered using CDIM. The highly discriminating
terms obtained are then recommended as tags for the test posts. A strategy is also implemented
in preprocessing step to handle missing data using information present online. Recommendation
results show a significant improvement over baseline. A self-optimizing version of this CDIM based
tag recommendation system is also presented. Self-optimization is achieved using a nonlinear opti-
mization model whose solution yields the clustering parameters that maximize the recommendation
accuracy within an administrator specified time window. Tag recommendation based on CDIM is
also discussed in the context of Self-Calibration where performance is improved using minimum
tweaking and error-correction on individual cluster level.

The second algorithm, Seq-Clust, is designed for data in form of sequences e.g. Web navigation
paths. It is an adaptation of \(k\)-modes clustering algorithm [Huang (1998)] with special features
e.g. giving more importance to the items occurring earlier in the sequence. The initial cluster
representatives are selected from the set of most dissimilar sequences which also fixes the number
of clusters \(k\). Seq-clust is applied to the problem of Web navigation path prediction. This problem
is famous in the area of Web usage mining and possesses a high commercial value for a list of reasons top of which is targeted marketing. We build generative as well as discriminative models to predict the first three page categories visited in a Web navigation path. Our generative models are based on Markov chain and Bayesian classification, whereas discriminative models are built using Support Vector Machine (SVM) [Joachims (1999)]. Seq-Clust is used to partition the Web navigation data into clusters of surfing behavior. Impact of this behavior clustering on Web surfer behavior prediction is then studied. Results show that Seq-Clust is successful in partitioning the Web navigation data into clusters whose representatives can be used to achieve high prediction efficiency. The impact of behavior clustering on accuracy of surfing prediction is not significant although.

Next two sections summarize our major and minor contributions respectively.

1.4.1 Major Contributions

Following are the major contributions of our work:

1. An efficient iterative partitional document clustering method CDIM (Clustering via Discrimination Information Maximization) is presented.
   - Objective is to maximize the sum of discrimination information provided by documents.
   - Resulting clusters are describable by their highly discriminating terms.
   - Feature selection for clustering is also provided in CDIM.

2. A content-based tag recommendation system for social bookmarking systems using CDIM is implemented.
   - Self-optimizing version of the content-based tag recommender is also presented.
   - We also present Self-Calibration and how can it be used to achieve self management? A self-calibrating tag recommendation system is proposed.
   - The role of personalization in recommending tags is studied. This study covers three graph based and one content based tag recommenders. We also discover and analyze three stages of users’ life cycle; beginners, followers and leaders.
3. We build generative and discriminative models for Web surfer behavior prediction and discover navigation patterns.

- Generative models are based on Bayes rule and Markov chain.
- Discriminative models are based on Support Vector Machine (SVM).
- The navigation patterns considered include pages (or page categories) visited in first N positions, type of visit (short or long), and range of page categories visited in first N positions.

4. A sequence clustering algorithm Seq-Clust is proposed.

- Seq-Clust is an adaptation of \( k \)-modes clustering algorithm.
- A new similarity measure that gives greater weight to matches at the beginning of the sequence is applied.
- The initial cluster representatives are selected from the set of most dissimilar sequences fixing the number of clusters.
- We study the impact of clustering Web navigation history before behavior prediction and observe that behavior clustering is good for efficient predictions. However, insignificant impact is discovered on prediction accuracy.

5. Evaluation

- Comparison of CDIM is done with five popular clustering methods on ten standard datasets with significance tests.
- Web surfing behavior prediction is evaluated on two real world datasets.
- Behavior clustering of two datasets is done through Seq-Clust.

1.4.2 Minor Contributions

We now phrase the minor contributions of our work.

1. A survey of document clustering, Web usage mining and tag recommendation techniques is conducted.
2. Hierarchical version of CDIM is implemented.

3. We use the discrimination measures relative risk and MDI as well as relevance measures domain relevance and domain consensus with CDIM.

4. Visualization of CDIM clusters in discrimination space is presented.

5. We study the impact of timestamp on Web surfing behavior prediction. Following important results are obtained. Incorporating timestamp in prediction models result in significant increase in time and space complexity of models. Whereas change in prediction accuracy is insignificant.

1.4.3 Related Publications

Parts of the research work presented in this thesis have been published in different venues mentioned below:


10. “CDIM: Clustering and Understanding Documents by Discrimination Information Maximization”, Malik Tahir Hassan, Asim Karim; IEEE Transactions on Knowledge and Data Engineering (TKDE), manuscript under review, 2013.

1.5 Thesis Outline

Chapter 1 provided the introduction and motivation for Web mining and the challenges involved in developing algorithms for it. Our contribution and related publications are also listed. Chapter 2 gives the background and related work in literature. Methods for clustering and Web mining are under focus of chapter 2. Chapter 3 presents our novel clustering algorithm CDIM (Clustering via Discrimination Information Maximization). Strengths of CDIM in comparison to other documents
clustering methods are established. Chapter 4 demonstrates a practical example of CDIM for Web mining by constructing a tag recommendation system based on discriminative clustering. Chapter 5 further extends our work on tag recommendation systems by presenting a self-optimizing tag recommender version. In addition, the idea of self-calibration is introduced and a self-calibrating tag recommender system is also discussed. Furthermore, Our study on the role of personalization in recommending tags is also presented in chapter 5. Chapter 6 presents a popular problem in Web usage mining “Web surfing behavior modeling and prediction”, and our generative and discriminative solutions to this problem. With the motivation of studying the impact of behavior clustering prior to prediction of Web navigation paths, we cluster the navigation history using our algorithm, Seq-Clust. The algorithm and the impact of behavior clustering on surfing prediction are also presented in chapter 6. We conclude in chapter 7 and point out promising future research directions.
Chapter 2

Background and Related Work

This chapter provides the relevant background and literature survey for Web mining and clustering. Web mining and its types are described. Since we focus on algorithms for Web content mining and Web usage mining types of Web mining, the related work in these areas is covered. Literature survey for the problem of Web surfing behavior prediction is also reported along with the work on sequence clustering. Related work for tag recommendation systems is also given. A description of clustering and its fundamental methods is presented followed by a survey of different document clustering techniques. As we contribute a discriminative clustering algorithm, use of discrimination information in the area of data processing is also covered.

Next section provides the fundamentals of Web mining and a study on related research work.

2.1 Mining the World Wide Web

The World Wide Web is a large data repository that allures the data mining practitioners a lot. Huge amounts of data getting in every second provide a great venue to test your knowledge extraction skills. Before moving to the interesting techniques of mining useful knowledge out of this knowledge gold-mine, lets have a look on the basics of Web and its evolution.
2.1.1 Evolution of the Web

Web 1.0: the traditional Web

In traditional Web, also called Web 1.0 the pages were static mostly. Authors of the Web pages used to create and publish their contents. Feedback from readers was not possible unless the authors share their personal contact information like phone number or email etc. Similarly, interaction of readers of a page with each other was also not supported. Although the objective of publishing information was achieved, yet it required a significant amount of software skills and money. There was a small number of Web page creators and most of the users were content consumers [Cormode and Krishnamurthy (2008)]. Dedicated webmasters were allowed to edit the content only and there was not any notion of user generated content.

Web 2.0: the Web as Platform

Web 2.0 is the network as platform, spanning all connected devices [Lewis (2006), Oreilly (2007)]. Making applications that exploit the “architecture of participation” and “collective intelligence” is the theme of Web 2.0. That means, delivering services that are updated continuously as more people use them. Sites like Wikipedia, Facebook, Fickr and BitTorrent are good demonstrations of Web 2.0 principles. The key principles [Oreilly (2007)] are summarized below.

1. scalable services instead of packaged software
2. controlled data sources that get richer as more people use them
3. trusting users as co-developers
4. harnessing collective intelligence
5. leveraging the long tail through customer self-service i.e. don’t ignore the small contributors as they are big in number.
6. software above the level of a single device
7. lightweight user interfaces, development models, and business models.
In short, Web 2.0 enables common users to connect to the Web as well as with each other, and they can now share comments and upload user generated content.

**Web 3.0: the Semantic and Personalized Web**

Although Web 2.0 resolved many previous issues on World Wide Web, but there is still more work to do. The distributed nature and easier sharing of information of Web 2.0 systems demand more of the user’s time and energy to remain updated and keep others updated too. Options are increased, so more human efforts are required in thinking and deciding. We need agents. Not humans, but machines. Machine agents that can create, request and deliver exact information on our behalf. Web 3.0 focuses on development of such Web solutions. The key components of Web 3.0 are the semantic Web and personalization [Agarwal (2009)]. The semantic Web tries to build self-managing softwares, machine interpretable knowledge bases and ontologies, accurate natural language processors etc. whereas personalization part focuses on facilitating individualized information needs.

**2.1.2 Web Mining**

Applying data mining techniques to World Wide Web is called Web Mining. In this thesis, the term Web will encompass all ages of World Wide Web i.e. Web 1.0, Web 2.0 and Web 3.0. The Web is a huge information center providing historical as well as up-to-the-minute data in many forms like text, audio, video etc. linked together for ease of access and understanding. Mining knowledge from this huge repository is highly desired and well rewarded effort.

**2.1.3 Types of Web Mining**

Web mining is generally divided into three branches namely Web Content Mining, Web Structure Mining and Web Usage Mining [Kosala and Blockeel (2000)]. This thesis concentrates on Web content and Web usage mining focusing on textual data.
Web Content Mining

Text classification, text clustering, finding frequent sub-structures etc. are some applications in the area of Web Content Mining. Input data is in the form of (hyper)text documents usually mapped to a bag-of-words, n-grams, ontology or relational representation. Data Mining is done with the help of different statistical, NLP based and machine learning techniques.

Web Usage Mining

The Web Usage data is generally in the form of log files generated automatically by the web servers. A Web server usually registers a log entry for every access of a Web page. It includes the URL requested, the IP address from which the request originated, and a timestamp. The Web usage data can also be collected with the help of cookies and tools that record users’ web sessions.

Web users definitely exhibit patterns of surfing behavior. Discovering such patterns have immense commercial value as the Web evolves into a primary medium for marketing and sales for many businesses. Web-based businesses seek useful users patterns to help identify promising events, potential risks, and make strategic decisions.

Web Structure Mining

Social networks analysis and citation analysis inspired the research on Web Structure Mining. Study of incoming and outgoing links of a Web page can help in finding authoritative Web pages. HITS (Hyperlink-Induced Topic Search) [Kleinberg (1999)] and PageRank [Brin and Page (1998)] are examples of successful algorithms that exploit structure of Web pages in mining high quality information.

2.1.4 Decade of Web Mining

Web mining field is more than a decade old now [Spiliopoulou et al. (2012)]. Research in this area started formally with WEBKDD’99 workshop. Earlier goals of Web mining were preprocessing of Web data and extraction of navigation patterns [Cooley et al. (2000), Srivastava et al. (2000), Baumgarten et al. (2000)], click stream clustering and prediction [Fu et al. (2000), Mobasher et al. (2000)].
(1999)], managing server load [Lan et al. (2000)], online advertising [Lee et al. (2001)] and privacy preservation [Broder (2000)] etc.

Web mining saw a rapid growth and around the middle of the first decade of new millennium, interesting new problems were on board. Recommendation systems [DeLong et al. (2005)], personalized applications [Kim and Chan (2005)], Web security [Mobasher et al. (2006)], clickstream mining [Lu et al. (2005)], semantic Web [Berendt (2006)] and adaptive applications [Sugiyama et al. (2004)] were the focus of Web mining researchers.

The new perception about Web is now as a social tool. Besides exchanging information with Web servers, individuals can interact with each other. Every one can share information and upload data. The term ‘social semantic Web’ has been under focus for some years [Mikroyannidis (2007)] which means a combination of social Web with semantic Web. Social network analysis [Ellison et al. (2007), De Nooy et al. (2011), Lewis et al. (2012)], sentiment analysis and opinion mining [OConnor et al. (2010), Amiri and Chua (2012)], efficient and accurate recommenders [Leginus et al. (2012)], private and secure mining [Berendt (2012)], semantic Web with machine usable knowledge bases [Rettinger et al. (2012)], personalized search [Leung et al. (2012)] and display advertising [Lang et al. (2012)] are key research topics under attention of Web mining community now a days.

2.1.5 Tag Recommendation Systems

Folksonomies and Tagging

Folksonomies are the online social communities where people share their content such as books, articles, pictures and videos etc. Examples of such systems are Flickr1, delicious2 and YouTube3. These Folksonomies have a common structure of user assigning tags to some items. Items could be either images, videos, articles or some bookmarks. Folksonomies provide this facility to user to tag or label their items while posting on the web. Tagging resources with one or more words or terms is a common way of organizing, sharing, and indexing information. Automatic tag recommendation for these applications can improve the organization of the information through ‘purposeful’ tag

\[\text{http://www.flickr.com}\]
\[\text{http://www.delicious.com}\]
\[\text{http://www.youtube.com}\]
recommendations. Moreover, automatic tag recommendations ease the task of users while posting new resources.

**Social Bookmarking Systems**

Social bookmarking systems e.g. Bibsonomy⁴, have become popular in recent years for organizing and sharing resources on the Web. Such systems allow users to build a database of resources, typically Web pages and publications, by adding basic information (such as URLs and titles) about them and by assigning one or more keywords or tags describing them. The tags serve to organize the resources and help improve recall in searches. Individual users databases are shared among all users of the system enabling the development of an information repository which is commonly referred to as a folksonomy.

**Tag Recommendation Approaches**

**Content based** tag recommendation and **collaborative filtering** based tag recommendation are two popular approaches. Content based Recommendation takes the knowledge from the attributes and properties of the user and items such as title or description of the post or image. Whereas in collaborative filtering a user is suggested what other users with similar interest are tagging. **Personalized approaches** give more weight to user’s personal preferences. This personal selection seems reasonable as user choice changes over time. Personalization follows an assumption that history repeats itself i.e. a user is likely to reuse tags from his history.

**Survey of Automatic Tag Recommendation Systems**

Many approaches proposed for tag recommendation assumed that either the user posting the resource and/or the resource itself has been seen in the historical data available to the system [Jäschke et al. (2007), Sigurbjörnsson and Van Zwol (2008), Jäschke et al. (2008), Symeonidis et al. (2008)]. If this is not the case, then only the contents of the posted resource can be relied upon. For social bookmarking systems, contents of resources are textual in nature requiring appropriate text and natural language processing techniques.

⁴http://www.bibsonomy.org
Several methods have been proposed for content-based tag recommendation in social bookmarking systems. Lipczak’s method extracts the terms in the title of a post, expands this set of terms by using a tag co-occurrence database, and then filters the result by the poster’s tagging history [Lipczak (2008)]. He reports significant improvements in performance after each step of this three step process. In [Lipczak et al. (2009)], Lipczak et. al. use resource IDs, resource contents, and user profiles to recommend tags. Symeonidis et al. [Symeonidis et al. (2010)] present a framework for tag recommendation based on semantic analysis. They represent the three entities in a social network (users, items, and tags) by 3-order tensors and apply Higher Order Singular Value Decomposition (HOSVD) to obtain a compact tagging model.

A key challenge in tag recommendation is dealing with sparsity of information. In a typical collaborative tagging system, the vast majority of tags are used very infrequently making learning tagging behavior very difficult.

Non-personalized tag recommendation algorithm like [Heymann et al. (2008), Song et al. (2008)] tend to repeat same tags for some particular items i.e. always popular tags are recommended for all users. A personalized recommendation completely avoids popular and for each item more preference to the user history is given.

Neighborhood models [Budura et al. (2009)] base their prediction on users having same interests. In this approach, as user evolves and changes her mind, topic drift can put limitations. Time Variant models for rating recommendation by koren et al. [Koren (2009), Cremonesi et al. (2010)] have shown good results. By our knowledge, no such time variant model is presented for tag recommendation dataset. This involves a complete data analysis to find out factor working for all data sets. Overall, neighborhood models may vary in terms of personalized or non-personalized recommendations. A good social group is formed because of high similarity in topics and if tags are suggested by filtering via personal choice, we can treat it like a personalized recommender.

Adapted PageRank [Hotho et al. (2006)] is an example of non-personalized tag recommender system. Hotho et. al adapted the famous PageRank algorithm to folksonomy problem.

Latent Factor models such as PITF [Rendle and Schmidt-Thieme (2010)] have shown to outperform Adapted PageRank. They rank tags by preferring tags a user have used then those she
has not used. In simple words, more weight for personal tag are given to each user to find per-
sonal choice to tags. It follows the assumption that most of the times tags originate from the user
history. Krestel et al. [Krestel and Fankhauser (2010)] have used probabilistic language models in
combination with Latent Dirichlet Allocation on Delicious dataset. It is found that personalization
with language model [Krestel and Fankhauser (2011)] outperforms top approaches like FolkRank
[Hotho et al. (2006)].

Hassan et al. [Hassan et al. (2009b)] and [Hassan et al. (2010)] present a content based tag
recommender that uses discriminative clustering to group the posts. Tags are then obtained from
the ranked terms of these clusters. Incorporating user’s personal history improves prediction per-
formance of this system.

Garg et al. [Garg and Weber (2008)] study the problem of personalized, interactive tag recom-
mandation for Flickr data set. They divided the users into small user, medium user and big user on
the basis of posts. A hybrid personalized algorithm works best for each type of user. The threshold
for user is based on number of post by users.

In a recent work, Song et. al. [Song et al. (2011)] argue that user-centered tag recommendation
approaches do not work well in practice. They suggest a graph based and a clustering based
document-centric (content based) approach for better tag recommendation.

Wei et. al. [Wei et al. (2011)] incorporate ratings in the existing ternary models and present the
quaternary relationship among users, items, tags and ratings as a 4-order tensor. They pose the
problem of recommendation as a multi-way latent semantic analysis problem and present better
results.

2.1.6 Self-Optimization and Self-Calibration

Self-Optimization is one of the four main self star properties of autonomic systems. A self-optimizing
system continually seeks to improve its performance e.g. by tuning parameters, thresholds, data
structures etc. Self-optimization is desirable in large scale systems and has been used with success
in communication systems. In the domain of social networks, [Michlmayr and Cayzer (2007)]
and [Sharma and Bedi (2009)] present pheromone evaporation technique of ant colony and swarm
intelligence for personalization in tagging systems. Hassan et. al. [Hassan et al. (2010)] propose a clustering based self-optimizing tag recommendation system.

Self-Calibration [Javed et al. (2012)] is the ability of a system to perceive the need for and the ability to execute minimal tuning to bridge the gap between system’s world view and incoming information about the outside world. Self-calibration is considered an enabling property that leads to a self-optimizing and self-healing system. The term has also been used to refer to the procedure by which raw outputs of sensors are mapped to standardized units [Bychkovskiy et al. (2003)]. There are automated methods for calibration but the process is explicitly initiated by the user deploying the system and is done once for an environment [Roth et al. (1987), Zhang and Huang (2006)].

2.1.7 Web Surfing Behavior Prediction and Clustering

Web surfer behavior modeling and prediction has been studied extensively in the literature. Here, we restrict our discussion to works that use navigation path information only. A majority of such works adopt probabilistic methods for Web surfer behavior prediction [Borges and Levene (2000), Manavoglu et al. (2003), Deshpande and Karypis (2004), Awad et al. (2008), Hassan et al. (2007)]. Borges and Levene [Borges and Levene (2000)] model navigation paths as N-gram probabilistic grammars which assume that the probability of visiting a Web page depends on the previous N pages visited. The parameter N has to be pre-specified or determined experimentally. Deshpande and Karypis [Deshpande and Karypis (2004)] propose the use of all kth order Markov models and suggest that many of the non-affecting states in a kth-order Markov model can be pruned to reduce the state space complexity. Eirinaki et al. [Eirinaki et al. (2005)] combine Markov models with link analysis methods for Web surfing predictions. In many problem settings, the link structure of the pages is not known and thus cannot be used in prediction models. Awad et al. [Awad et al. (2008)] combine Markov models and SVM using Dempster’s rule for surfing predictions. They incorporate domain knowledge also but do not study the impact of clustering the history. Markov mixture models combined with maximum entropy based learning is the solution proposed by Manavoglu et al. [Manavoglu et al. (2003)]. These probabilistic methods are overly complicated for predicting
navigation patterns of practical value like predicting the first three pages viewed by users. For this reason, we adopt Bayes rule and Markov chain models for prediction as done by Hassan et al. [Hassan et al. (2009a)].

Web data clustering has been employed for user segmentation, behavior visualization, and to a lesser extent, prediction. Lu et al. [Lu et al. (2005)] generate Significant Usage Patterns (SUP) by clustering clickstream data using a pair-wise alignment algorithm and then building a first-order Markov model for each of the clusters. A similar problem and solution is proposed by Halvey et. al. [Halvey et al. (2005)]. They use clustering based on time and show improvement in navigation prediction through an empirical analysis carried out on a sample of usage logs for Wireless Application Protocol (WAP) browsing. On the other hand, our approach focusses on key navigation patterns or behaviors and is based on behavior clustering (not time clustering). Pallis et al. [Pallis et al. (2007)] have also done clustering of users’ navigation into sessions but their focus is on identifying patterns and similarities, and not navigation path prediction. Sadikov et al. [Sadikov et al. (2010)] propose a clustering algorithm to improve the query suggestions of a search engine. They build a Markov model of users’ search behavior and their clustering model is complete link with cosine similarity measure. They focus on clustering the query refinements whereas focus of our work is clustering and prediction of navigation behaviors. Sequence based clustering of Web sessions has been done by Park et al. [Park et al. (2008)]. They investigate the impact of different sequence representations, similarity measures, number of clusters, number of web pages etc. on clustering performance. We on the other hand study the impact of clustering on navigation prediction models.

Niu et al. [Niu and Shiu (2005)] extend the Markov model approach by introducing a similarity measure for navigation paths. The similarity measure is used to group Web users based on their access logs. Predictions are then done based on the historical records in a single group. Our approach uses a new similarity measure and a simpler prediction model. Pena et al. [Cadez et al. (2003)] have developed a tool for visualizing navigation patterns using model based clustering. Their emphasis is not on prediction of future visits but on the visualization of behavior clusters. Another work focussing on behavior cluster visualization is reported by Kim [Kim (2007)].
similarity measure, which gives greater weight to matches at beginning of navigation paths, is similar to that used by us. Liu and Keselj [Liu and Keselj (2007)] cluster the user sessions to get representative navigation patterns and use them in combination with the contents of the web pages to classify and predict future requests. In our approach the contents of the web pages are considered to be unknown.

Li and Lee [Li and Lee (2009)] propose a single pass algorithm to find top-k path traversal patterns for click streaming data. Their algorithm, DSM-TKP (Data Stream Mining for Top-K Path traversal patterns), uses a summary data structure named TKP-forest (a forest of Top-K Path traversal patterns). The forest is maintained using a forest pruning mechanism. Pitman et al. [Pitman et al. (2010)] use clustering of users’ web sessions to extract business intelligence from web logs of a Swedish tourism portal. After applying Principle Component Analysis (PCA), they use x-means [Pelleg and Moore (2000)] to find that different user groups have significantly different information needs. Their focus is analysis instead of prediction.

After the background and survey of Web mining, we present selected relevant clustering literature in the following section.

2.2 Clustering

Clustering is the process of dividing objects into groups such that the objects in a group are very similar to one another but very dissimilar to the objects in other groups. The similarity is calculated using the values of the attributes that define the object. The similarity measure can be symmetric or asymmetric. It is symmetric if similarity of object X to object Y is same as similarity of object Y to object X.

2.2.1 Clustering Methods

Clustering methods are traditionally organized into the following categories: partitioning methods, hierarchical methods, density based methods, grid based methods, model based methods and constraint based methods [Han and Kamber (2006)].

The partitioning methods try to reach a partitioning of the data set that optimizes an objective
function e.g. minimize squared error. Unfortunately, these methods either require the number of partitions \( k \), or \( k \) initial centroids (seeds) to start the algorithm. This initialization issue has been focused in [Dubes (1987)]. However, as compared to other methods, partitioning methods work faster on large data sets and usually perform better for spherical shaped clusters due to distance based similarity calculation mostly. \( k \)-means [Lloyd (1982), MacQueen (1967)] and \( k \)-medoids [Kaufman and Rousseeuw (1990)] are two popular partitioning methods.

Hierarchical clustering methods like BIRCH [Zhang et al. (1996)] and CURE [Guha et al. (1998)] etc. work by grouping data objects into a tree of clusters. They have a problem of inability to correct the erroneous decisions once a merge or split action is done.

Density based methods are used to discover clusters of arbitrary shape. They grow the size of a cluster by counting the number of objects in its neighborhood e.g. DBSCAN [Ester et al. (1996)].

Grid based methods quantize the object space into a finite number of cells that form a grid structure. All of the clustering operations are performed on the grid structure. STING [Wang et al. (1997)] and CLIQUE [Agrawal et al. (1998)] are two popular grid based clustering methods.

Model based clustering methods attempt to optimize the fit between the given data and some mathematical model. These methods can follow a statistical approach e.g. COBWEB [Fisher (1987)] or a neural network approach e.g. SOM [Kohonen (2001)].

A Constraint based clustering method [Tung et al. (2001), Basu et al. (2004)] groups objects based on application dependent or user-specified constraints for example these pair of objects can not belong to same cluster.

Clustering can be hard or soft. In hard clustering, an object is assigned to only one cluster. While in soft clustering, an object can be assigned to more than one cluster with some certainty values. EM [Dempster et al. (1977)] and Fuzzy c-means [Bezdek (1973)] use the soft clustering scheme.

Spectral clustering is another popular clustering method used to cluster high dimensional data objects by using eigenvalues of the similarity matrix for reducing the dimensions first. Shi-Malik [Shi and Malik (2000)] algorithm is an example of spectral clustering technique. In contrast to \( k \)-means spherical shape clusters, spectral clustering has no strong assumptions on the form of
clusters and can group more general data like intertwined spirals. However, spectral clustering methods are sensitive to the selection of similarity graph of data objects [von Luxburg (2007)].

For high dimensional data, other common clustering approaches are subspace clustering, projected clustering, correlation clustering and pattern based clustering [Kriegel et al. (2009b)]. Subspace clustering tries to find all clusters in all subspaces of the data. Subspaces may be axis aligned or affine. Clustering of axis aligned subspaces is also called projected clustering. However, in general, these projections may not be aligned to any axis. Correlation clustering uses the relationships between objects instead of their actual representations only, while clustering. Pattern based approaches exploit the patterns or common behavior of objects in the data sets. Bi-clustering or co-clustering or two-mode clustering is a pattern based clustering method where rows and columns of the data matrix are clustered simultaneously. Co-clustering has been successfully employed in gene expression analysis and document clustering for example [Dhillon et al. (2003)].

2.2.2 Document Clustering

Data clustering has been studied extensively for over 50 years with widespread usage in applications across different disciplines [Jain (2010), Jain et al. (1999)]. This popularity stems from the intuitive objective of clustering; finding groups of related objects. Nonetheless, developments in data clustering continue specially in applications areas like document clustering. In general, document clustering can be partitional or hierarchical in nature [Tan et al. (2006)]. Partitional clustering finds $K$ clusters (where $K$ is an input parameter) by optimizing an objective function that captures an appropriate notion of clustering, while (agglomerative) hierarchical clustering creates a tree of clusters by performing local merge operations. Since our method is partitional in nature, we will focus primarily on this type of clustering methods.

Content-based document clustering has also been studied since the 1960s [Salton (1963), Andrews and Fox (2007)]. Document clustering continues to be a challenging problem because of (1) the high dimensionality of the term-document space, (2) the sparsity of the documents in the term-document space, and (3) the difficulty of incorporating appropriate term-document semantics for improved clustering quality and understandability. Moreover, real-world document clustering
often involves large document collections thus requiring the clustering method to be efficient.

The $K$-means algorithm continues to be popular for document clustering due to its efficiency and ease of implementation [Tan et al. (2006)]. It is a partitional clustering method that optimizes an objective function via iterative two-step procedure. Usually, documents are represented by terms that are weighted by term-frequency or term-frequency-inverse-document-frequency values, and documents are compared in the term space by the cosine similarity measure. Several clustering objective functions can be optimized [Zhao and Karypis (2001)]. These functions can be based on notions of cohesion, separation, or both cohesion and separation (hybrid) of clusters. It has been reported that the objective function of the traditional $K$-means algorithm which maximizes the similarity of documents to their cluster mean produces reliable clusterings [Zhao and Karypis (2001)]. The Repeated Bisection clustering method, which splits clusters into two until the desired number of clusters are obtained, has been shown to produce better clusterings especially when $K$ is large (greater than 20) [Steinbach and Karypis (2000)]. These $K$-means based methods are efficient and accurate for many practical applications. Their primary shortcoming is poor interpretability of the clusters where the cluster mean vector is often not a reliable indicator of the documents in a cluster.

Some researchers have used external knowledge bases to semantically enrich the document representation for document clustering [Hu et al. (2009), Zhang et al. (2007)]. In [Hu et al. (2009)], Wikipedia’s concepts and categories are adopted to enhance the document representation, while in [Zhang et al. (2007)] several ontology-based (e.g. WordNet) term relatedness measures are evaluated for semantically smoothing the document representation. In both works, it has been shown that the quality of clusterings produced by the $K$-means algorithm improves over the baseline (“bag of words”) document representation. However, extracting information from knowledge bases is computationally expensive. Furthermore, these approaches suffer from the same shortcomings of $K$-means regarding cluster understandability.

The challenge of high dimensional data clustering, including that of document clustering, has received much interest in recent years [Kriegel et al. (2009a)]. These methods try to find clusters in lower dimensional subspace(s) of the original term space or a transformed space. One way to
find clusters in transformed spaces is through Non-Negative Matrix Factorization (NMF). NMF approximates the term-document matrix by the product of term-cluster and document-cluster matrices [Xu et al. (2003)]. Extensions to this idea, with the goal of improving the interpretability of the extracted clusters, have also been proposed [Xu and Gong (2004), Cai et al. (2011)]. Nonetheless, these methods are restricted by their focus on approximation rather than semantically useful clusters. Our method, on the other hand, focuses on finding highly discriminating clusters where documents having similar behaviors (discrimination score for a cluster) are grouped together.

Another way to find clusters in transformed spaces is to combine clustering methods with dimensionality reduction techniques. Unsupervised dimensionality reduction techniques have been proposed for document clustering, and a comparison of four such techniques is given in [Tang et al. (2005)]. It is reported that when Latent Semantic Indexing (LSI) is combined with $K$-means algorithm, reliable clusterings are obtained. The combination of $K$-means with Linear Discriminant Analysis (LDA), which is a supervised dimensionality reduction technique, has also been investigated [De la Torre and Kanade (2006), Ding and Li (2007)]. These methods iteratively perform clustering in low-dimensional spaces found by LDA. Our method uses a similar iterative procedure of finding clusters in a low-dimensional space. However, its procedure is more efficient than the methods that use LDA and LSI.

Co-clustering [Dhillon et al. (2003)] in the area of text mining is a method of clustering the documents and terms simultaneously at all stages. The result is clusters of documents that show similar behavior across clusters of terms, or vice versa. In comparison, CDIM [Hassan and Karim (2012)] is by nature one-way clustering that besides clustering the documents, also returns a ranking of terms in every cluster. Moreover, this ranking of terms by CDIM is not hard, i.e. same terms can appear in more than one clusters.

In addition to the discriminative and statistical clustering methods stated above, there are generative techniques present in the literature. Latent Dirichlet Allocation (LDA) [Blei et al. (2003)] and other related methods are applied popularly for document clustering, topic modeling and summarization purposes [Li et al. (2012), Xu and Zhu (2013), Liao et al. (2013)].
2.2.3 Discrimination Information

Discrimination, or its opposite association, is a fundamental concept in information processing. It is central to many data mining tasks such as classification and feature selection. Measures of discrimination information come from statistics and information theory. Common measures include relative risk, odds ratio, risk difference, information gain, and Kullback-Leibler divergence. These measures are corpus-based, i.e., they are estimated from a data collection.

In recent years, there has been growing interest in using statistically sound measures in data mining [Li et al. (2005; 2007)]. In the biomedical domain, on the other hand, measures like relative risk and odds ratio have been used for a long time for cohort studies and factor analysis [Hsieh et al. (1985), LeBlanc and Crowley (1992)]. In text processing, such measures have been used primarily for feature selection [Chung and Lee (2001)]. More recently, measures like relative risk and information gain have been used to quantify the discrimination information provided by terms for text classification purposes [Junejo and Karim (2008), Malik et al. (2011), Junejo and Karim (2013)]. These works highlight the suitability of building learning models from term discrimination information. Our method uses the relative risk [Li et al. (2005)] or MDI [Cai and Van Rijsbergen (2009)] measures to quantify the discrimination information provided by terms and develops a transformation based on this information for document clustering.

2.3 Conclusion

We provide background and survey of relevant literature in this chapter. Areas covered broadly include Web mining and clustering. In Web mining the focus is on Web usage mining including Web surfer behavior prediction and automatic tag recommendation, and Web content mining including text document clustering. As we propose a discriminative clustering algorithm for text documents, we also cover the use of discrimination information in our survey.

The next chapter focuses on the Web content mining through document clustering and we present in detail our discriminative clustering method for text documents.
Chapter 3

Discriminative Clustering for Text

Previous chapters suggest the application of clustering for the purpose of Web content and Web usage mining feasible. This chapter focuses on the first aspect in detail i.e. Web content mining through clustering. Although Web content can be in other forms including audio, video, graphics etc. but the major component is normally text. A novel document clustering algorithm CDIM (Clustering via Discrimination Information Maximization) [Hassan and Karim (2012)] is presented here that can be applied successfully to group text contents into understandable clusters.

3.1 Introduction

Data clustering is one of the most widely used task in data mining due to its capability for summarizing large data collections and its simplicity of purpose. The objective of data clustering methods is to find groups of data objects that are related to one another within a group and are unrelated to objects in other groups. These methods, which are unsupervised in nature, often optimize an objective function that captures an appropriate notion of clustering (e.g. maximize similarity between objects within groups).

Text document clustering discovers groups of related documents in large document collections. Its importance has grown significantly over the years as the world moves toward a paperless environment and the Web continues to dominate our lives. Efficient and effective document clustering methods can help in better document organization (e.g. digital libraries, corporate documents, etc)
as well as quicker and improved information retrieval (e.g. online search).

Besides the need for efficiency, document clustering methods should be able to handle the large term space of document collections to produce readily understandable clusters. These requirements are often not satisfied in popular clustering methods. For example, in $K$-means clustering [Jain et al. (1999)], documents are compared in the term space, which is typically sparse, using generic similarity measures without considering the term-document semantics other than their vectorial representation in space. Moreover, it is not straightforward to interpret and understand the clusters formed by $K$-means clustering; the similarity of a document to its cluster’s mean provides little understanding of the document’s context or topic.

In this chapter, we present a new document clustering method based on discrimination information maximization (CDIM). Iterative procedure of CDIM repeatedly projects documents onto a $K$-dimensional discrimination information space and assigns documents to the cluster along whose axis they have the largest value. The discrimination information space is defined by term discrimination information estimated from the labeled document collection produced in the previous iteration. This procedure maximizes the sum of discrimination information provided by all documents. A key advantage of using term discrimination information is that each cluster can be identified by a list of highly discriminating terms. These terms can also be thought of as units of thought describing a cluster in the document collection. As a result of this semantic interpretation, the clusters produced by CDIM are understandable by their discriminating terms. Since CDIM is posed as an optimization problem, there is room to apply different measures in objective function. We present results using Relative Risk (RR) [Li et al. (2005)], Measurement of Discrimination Information (MDI) [Cai and Van Rijsbergen (2009)], Domain Relevance (DR) and Domain Consensus (DC) [Navigli and Velardi (2004)]. Other variations of CDIM that we implement include CDIM using repeated bisection. We evaluate the performance of CDIM on ten popular text data sets. In clustering quality evaluation, CDIM is found to produce high quality clusters that are better than those produced by non-negative matrix factorization (NMF) and comparable to those produced by the best $K$-means variants. We determine clustering quality by calculating the value of BCubed F-measure [Amigó et al. (2009)] which combines BCubed precision and BCubed recall. Our results
suggest the practical suitability of CDIM for clustering and understanding of document collections.

The rest of the chapter is organized as follows. Our document clustering method, CDIM, is described in detail in Section 3.2. Section 3.4 presents our experimental setup. Section 3.5 discusses the results of our experiments. We conclude in Section 3.6.

3.2 CDIM – Our Document Clustering Method

CDIM (Clustering via Discrimination Information Maximization) is an iterative partitional document clustering method that finds \( K \) groups of documents in a \( K \)-dimensional discrimination information space transformed from the \( M \)-dimensional input space. It does this by following an efficient two-step procedure (similar to that of the \( K \)-means algorithm) of document projection and assignment with the goal of maximizing the sum of documents’ discrimination scores. CDIM’s clusters are describable by highly discriminating terms related to the context/topic of the documents in the cluster. We start our presentation of CDIM by formally stating the problem.

3.2.1 Problem Statement

Let \( X = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{M \times N} \) be the term-document matrix in which the \( i \)th document \( x_i = [x_{1i}, x_{2i}, \ldots, x_{Mi}]^T \) is represented by an \( M \)-dimensional vector (\( i \)th column of matrix \( X \)). \( M \) is the total number of distinct terms in the \( N \) documents. The weight of term \( j \) in document \( i \), denoted by \( x_{ji} \), is equal to the count of term \( j \) in document \( i \).

Our goal is to find \( K \) (usually in practice \( K \ll \min\{M, N\} \)) clusters \( C_k \ (k = 1, 2, \ldots, K) \) of documents such that if a document \( x \in C_k \) then \( x \not\in C_j, \forall j \neq k \). Thus, we assume hard partitioning of the documents among the clusters; however, this assumption can be relaxed trivially in CDIM but we do not discuss this further in our current work. In addition to the cluster composition, we will also like to find significant describing terms for each cluster. Let \( T_k \) be the index set of significant terms for cluster \( k \).
3.2.2 Clustering Objective Function

CDIM finds $K$ clusters in the document collection by maximizing the sum of discrimination scores of documents for their respective clusters. If we denote the discrimination information provided by document $i$ for cluster $k$ by $d_{ik}$ and the discrimination information provided by document $i$ for all clusters but cluster $k$ by $\bar{d}_{ik}$, then the discrimination score of document $i$ for cluster $k$ is defined as $\hat{d}_{ik} = d_{ik} - \bar{d}_{ik}$. CDIM’s objective function can then be written as

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} r_{ik} \left( d_{ik} - \bar{d}_{ik} \right) \quad (3.1)$$

where $r_{ik} = 1$ if document $i$ is assigned to cluster $k$ and zero otherwise. Document discrimination information ($d_{ik}$ and $\bar{d}_{ik}$) is computed from term discrimination information that in turn is estimated from the current labeled document collection. These computations are discussed in the following subsections.

Intuitively, CDIM seeks a clustering in which the discrimination information provided by documents for their cluster is higher than the discrimination information provided by them for the remaining clusters. It is not sufficient to maximize just the discrimination information of documents for their respective clusters as they may also provide high discrimination information for the remaining clusters.

The objective function $J$ is maximized by using a greedy two-step procedure. In one step, given a cluster assignment defined by $r_{ik}, \forall i, k$, $J$ is maximized by estimating $d_{ik}, \forall i, k$ and $\bar{d}_{ik}, \forall i, k$ from the labeled document collection. This estimation, as discussed in the following subsections, is done using maximum likelihood estimation. In the other step, given estimated discrimination scores $\hat{d}_{ik}, \forall i, k$ of documents, $J$ is maximized by assigning each document to the cluster $k$ for which the document’s discrimination score is maximum. This two-step procedure continues until the change in $J$ from one iteration to the next drops below a specified threshold value.
3.2.3 Term Discrimination Information

The discrimination information provided by a document is computed from the discrimination information provided by the terms in the document. The discrimination information provided by a term is quantified with a measure of discrimination information that is estimated from the labeled document collection. The following subsections show our formulation for two discrimination measures: Relative Risk (RR)\cite{Li2005} and Measurement of Discrimination Information (MDI)\cite{Cai2009}.

**Term Discrimination Information Using Relative Risk**

The relative risk \cite{Li2005} of a term for cluster $k$ over the remaining clusters is used as its discrimination information for cluster $k$. Mathematically, the discrimination information of term $j$ for cluster $k$ and term $j$ for all clusters but $k$ is given by

$$
\begin{align*}
  w_{jk} &= \begin{cases} 
    \frac{p(x_j | C_k)}{p(x_j | \bar{C}_k)} & \text{when } p(x_j | C_k) - p(x_j | \bar{C}_k) > t \\
    0 & \text{otherwise}
  \end{cases} \\
  \bar{w}_{jk} &= \begin{cases} 
    \frac{p(x_j | \bar{C}_k)}{p(x_j | C_k)} & \text{when } p(x_j | \bar{C}_k) - p(x_j | C_k) > t \\
    0 & \text{otherwise}
  \end{cases}
\end{align*}
$$

where $p(x_j | C_k)$ is the conditional probability of term $j$ in cluster $k$ and $\bar{C}_k$ denotes all clusters but cluster $k$. The term discrimination information is either zero (no discrimination information) or greater than one with a larger value signifying higher discriminative power.

**Term Discrimination Information Using MDI**

\cite{Cai2009} study the semantic relatedness between terms using terms discrimination information and present the issue of MDI (Measurement of Discrimination Information). They suggest two discrimination measures $ifd_{I\Sigma}$ and $ifd_{I\Sigma}$, and their combined discrimination
These measures in CDIM context are defined as below:

$$\text{ifd}_{I1\Sigma}(x_j) = p(x_j|C_k) \log \frac{p(x_j|C_k)}{\lambda_1 p(x_j|C_k) + \lambda_2 p(x_j|\bar{C}_k)}$$  \hspace{1cm} (3.4)

$$\text{ifd}_{I2\Sigma}(x_j) = p(x_j|\bar{C}_k) \log \frac{p(x_j|\bar{C}_k)}{\lambda_1 p(x_j|C_k) + \lambda_2 p(x_j|\bar{C}_k)}$$  \hspace{1cm} (3.5)

and the combined discrimination measure becomes

$$\text{ifd}_K = \lambda_1 \text{ifd}_{I1\Sigma} + \lambda_2 \text{ifd}_{I2\Sigma}$$

$\lambda_1$ and $\lambda_2$ are the prior probabilities of $C_k$ and $\bar{C}_k$ respectively, and can be set to 0.5 to get a symmetric version of these discrimination measures.

Discussing the application of these measures, Cai and Rijsbergen [Cai and Van Rijsbergen (2009)] present following inequality to judge whether a term supports category 1 more than 2

$$\psi_1 = \lambda_1 \text{ifd}_{I1\Sigma} - \lambda_2 |\text{ifd}_{I2\Sigma}| > 0$$  \hspace{1cm} (3.6)

Similarly, the pre-condition for a term to support category 2 more than category 1 is

$$\psi_2 = \lambda_2 \text{ifd}_{I2\Sigma} - \lambda_1 |\text{ifd}_{I1\Sigma}| > 0$$  \hspace{1cm} (3.7)

Using inequalities 3.6 and 3.7, the discrimination term weights for CDIM algorithm, $w_{jk}$ and $\bar{w}_{jk}$ are calculated as below

$$w_{jk} = \begin{cases} 
\psi_1 & \text{when } \psi_1 > t \\
0 & \text{otherwise}
\end{cases}$$  \hspace{1cm} (3.8)

$$\bar{w}_{jk} = \begin{cases} 
\psi_2 & \text{when } \psi_2 > t \\
0 & \text{otherwise}
\end{cases}$$  \hspace{1cm} (3.9)

The conditional probabilities are estimated via maximum likelihood estimation with Good-
Turing smoothing [Chen and Goodman (1999)]. Smoothing is summarized in next section.

3.2.4 Smoothing

Smoothing is performed to avoid zero-probabilities when estimation is done using maximum likelihood method. Higher probabilities are reduced slightly to make room for lower ones. Besides prevention of zero-probabilities, smoothing is also known to improve modeling. There exist many smoothing techniques with different level of complexity in literature [Chen and Goodman (1999)]. Jelinek-Mercer, Katz, Witten-Bell and Kneser-Ney are some of the well-known smoothing techniques to name. These combine the higher order n-gram models with lower order n-gram models for improved probability estimation. Additive smoothing and Good-Turing smoothing are two simpler techniques that do not combine the higher order models with lower order models. Additive smoothing has been criticized and performs poorly many times. As CDIM algorithm depends on unigrams only and there are no higher order models involved, we present results when probabilities are estimated using the simplest Good-Turing estimator [Kochanski (2006)]. So the non-zero probabilities are reduced by a factor of $1 - \frac{E(1)}{T}$ and zero probabilities get equal share from $\frac{E(1)}{T}$. Here $E(1)$ is the expected number of terms that occur once and $T$ is the corpus size.

3.2.5 Relatedness of Terms to Clusters

In Equations 3.2, 3.3, 3.8 and 3.9, $t \geq 0$ is a term selection parameter that controls the exclusion of terms that provide insignificant discrimination information. As the value of $t$ is increased from zero, fewer terms will have a high discrimination information. The index set of terms that provide significant discrimination information for cluster $k$ ($T_k$) is defined as

$$T_k = \{ j | w_{jk} > 0, \forall j \}.$$  \hspace{1cm} (3.10)

These terms and their discrimination information provide a good understanding of the context of documents in cluster $k$ in contrast with those in other clusters in the document collection. In general, $T_k \cap T_j \neq \emptyset, \forall j \neq k$. That is, there may be terms that provide significant discrimination information for more than one cluster. Also, depending on the value of $t$, there may be terms that
do not provide significant discrimination information for all clusters.

In a study discussed in [Morris and Hirst (2004)], it has been shown that humans comprehend text by associating terms with particular contexts or topics. These relationships are different from the traditional lexical relationships (e.g. synonymy, antonymy, etc), but are more fundamental in conveying meaning and understanding. Recently, it has been shown that the degree of relatedness of a term to a context is proportional to the term’s discrimination information for that context in a corpus [Cai and Van Rijsbergen (2009)]. Given these studies, we can consider all terms in $T_k$ to be related to cluster $k$ and the strength of this relatedness is given by the term’s discrimination information. This is an important characteristic of CDIM whereby each cluster’s context is describable by a set of related terms. Furthermore, these terms and their weights (discrimination information) define a $K$-dimensional space in which documents are comparable by their discrimination information.

### 3.2.6 Document Discrimination Information

A document $i$ is describable by the terms it contains. Each term $j$ in the document vouches for the context or cluster $k$ according to the value of the term’s discrimination information $w_{jk}$. Equivalently, each term $j$ in the document has a certain degree of relatedness to context or cluster $k$ according to the value $w_{jk}$. The discrimination information provided by document $i$ for cluster $k$ can be computed as the average term discrimination information for cluster $k$:

$$d_{ik} = \frac{\sum_{j \in T_k} x_{ji} w_{jk}}{\sum_j x_{ji}}. \quad (3.11)$$

A similar expression can be used to define $\bar{d}_{ik}$. The document discrimination information $d_{ik}$ can be thought of as the relatedness (discrimination) of document $i$ to cluster $k$. The document discrimination score is given by $\hat{d}_{ik} = d_{ik} - \bar{d}_{ik}$; the larger this value is, the more likely that document $i$ belongs to cluster $k$. Note that a term contributes to the discrimination information of document $i$ for cluster $k$ only if it belongs to $T_k$ and it occurs in document $i$. If such a term occurs multiple times in the document then each of its occurrence contributes to the discrimination information. Thus, the discrimination information of a document for a particular cluster increases.
with the increase in occurrences of highly discriminating terms for that cluster.

3.2.7 Algorithm

CDIM can be described more compactly in matrix notation. CDIM’s algorithm, which is outlined in Algorithm 1, is described next.

Let \( \mathbf{W} \) (\( \bar{\mathbf{W}} \)) be the \( \times \) \( K \) matrix formed from the elements \( w_{jk}, \forall j, k \) (\( \bar{w}_{jk}, \forall j, k \)), \( \mathbf{\hat{D}} \) be the \( N \times K \) matrix formed from the elements \( \hat{d}_{ik}, \forall i, k \), and \( \mathbf{R} \) be the \( N \times K \) matrix formed from the elements \( r_{ik}, \forall i, k \). At the start, each document is assigned to one of the \( K \) randomly selected seeds using cosine similarity, thus defining the matrix \( \mathbf{R} \). Then, a loop is executed consisting of two steps. In the first step, the term discrimination information matrices (\( \mathbf{W} \) and \( \bar{\mathbf{W}} \)) are estimated from the term-document matrix \( \mathbf{X} \) and the current document assignment matrix \( \mathbf{R} \). The second step projects the documents onto the relatedness or discrimination score space to create the discrimination score matrix \( \mathbf{\hat{D}} \). Mathematically, this transformation is given by

\[
\mathbf{\hat{D}} = (\mathbf{X} \Sigma)^T (\mathbf{W} - \bar{\mathbf{W}}) \tag{3.12}
\]

where \( \Sigma \) is a \( N \times N \) diagonal matrix defined by elements \( \sigma_{ii} = 1/\sum_j x_{ji} \). The matrix \( \mathbf{\hat{D}} \) represents the documents in the \( K \)-dimensional discrimination score space.

Documents are re-assigned to clusters based on their discrimination scores. A document \( i \) is assigned to cluster \( k \) if \( \hat{d}_{ik} \geq \hat{d}_{ij}, \forall j \neq k \) (ties are broken arbitrarily). In matrix notation, this operation can be written as

\[
\mathbf{R} = \text{maxrow}(\mathbf{\hat{D}}) \tag{3.13}
\]

where ‘maxrow’ is an operator that works on each row of \( \mathbf{\hat{D}} \) and returns a 1 for the maximum value and a zero for all other values. The processing of Equations 3.12 and 3.13 are repeated until the absolute difference in the objective function becomes less than a specified small value. The objective function \( J \) is computed by summing the maximum values from each row of matrix \( \mathbf{\hat{D}} \).

The algorithm outputs the final document assignment matrix \( \mathbf{R} \) and the final term discrimination information matrix \( \mathbf{W} \).
Algorithm 1 CDIM – Document Clustering via Discrimination Information Maximization

Require: $X$ (term-document matrix), $K$ (no. of clusters)

1: $R^{(0)} \leftarrow$ initial assignment of documents to clusters
2: $\tau \leftarrow 0$
3: $J^{(0)} \leftarrow 0$
4: repeat
5: $\hat{W}^{(\tau)}$, $\bar{W}^{(\tau)} \leftarrow$ term discrimination info estimated from $X$ and $R^{(\tau)}$ (Eqs. 3.2 and 3.3)
6: $\hat{D}^{(\tau+1)} \leftarrow (X\Sigma)^T(W^{(\tau)} - \bar{W}^{(\tau)})$
7: $R^{(\tau+1)} \leftarrow \text{maxrow}(\hat{D}^{(\tau+1)})$
8: $J^{(\tau+1)} \leftarrow$ sum of max discrimination scores from each row of $\hat{D}^{(\tau+1)}$
9: $\tau \leftarrow \tau + 1$
10: until $|J^{(\tau)} - J^{(\tau-1)}| < \epsilon$
11: return $R$ (document assignment matrix), $W$ (term discrimination info matrix)

3.2.8 Key Characteristics of CDIM

We highlight some characteristics and properties of our document clustering method below.

1. It is easy to see that the computational time complexity of CDIM is $O(KMNI)$ where $I$ is the number of iterations required to reach the final clustering. Thus, the computational time of CDIM depends linearly on the clustering parameters. The advantage over other $K$-means variants is that comparisons are in $K$-dimensional space instead of $M$-dimensional documents, and $K$ (number of clusters) is normally significantly smaller than $M$ (number of terms).

2. CDIM does not require the specification of a document-to-document similarity or dissimilarity measure. Documents are projected onto an $K$-dimensional discrimination score space in such a way that the relevance of a document to a cluster is given by its value on the corresponding axis – the larger this value is, the more relevant the document is to that cluster.

3. In addition to outputting the document assignment matrix, CDIM also outputs the term discrimination information matrix ($W$). This matrix identifies the significant discriminating terms for each cluster and quantifies their discrimination information or relatedness for/to each cluster. This information is valuable for understanding the context of documents in each cluster.

4. The two-step procedure of document assignment based on discrimination information estimation from the labeled documents produced by the previous iteration represents a nave
semi-supervised learning approach [Xue and Weiss (2009)]. In [Xue and Weiss (2009)], this approach is shown to converge smoothly in about 20 iterations.

5. CDIM includes a natural way for term selection via the term selection parameter $t$ (see Equations 3.2 and 3.3). By increasing the value of $t$, terms providing little discrimination information (potential noise) can be removed easily to further improve clustering performance.

We illustrate and discuss some of these characteristics in our experiments i.e. section 3.4.

### 3.2.9 Convergence of CDIM

CDIM is a two-step iterative algorithm like K-means and Expectation-Maximization (EM). The two steps of CDIM are calculation of discrimination scores and assignment of documents to clusters. In K-means, a cluster is represented by its mean or centroid. Whereas in CDIM a cluster is represented by its significant discriminating terms. The iterations in K-means try to stabilize the cluster means; likewise, the iterations in CDIM try to stabilize the set of significant discriminating terms. Below, we prove that CDIM converges/terminates by proving that both of its steps monotonically increase the objective function, and the value of objective function is bounded from above.

**Theorem 1.** CDIM algorithm converges to local maxima.

**Proof.** Consider CDIM’s objective function (eq. 3.1). It is the sum of discrimination scores of documents for their respective clusters, where the discrimination score of document $i$ for cluster $k$ is $\hat{d}_{ik} = d_{ik} - \bar{d}_{ik}$. The objective function can be written as the sum of $K$ clusters’ documents’ discrimination scores:

$$J = \sum_{x_i \in C_1} r_{i1}(\hat{d}_{i1}) + \sum_{x_i \in C_2} r_{i2}(\hat{d}_{i2}) + \ldots + \sum_{x_i \in C_K} r_{iK}(\hat{d}_{iK})$$

(3.14)

Let $C_1, C_2, \ldots, C_k$ be our current clustering with objective function value $J_t$. Based on current labels we calculate discrimination scores and find that a document $x_n \in C_i$ needs relabeling, i.e.,

$$\hat{d}_{nj} > \hat{d}_{ni}, \forall i \neq j$$

(3.15)

42
This implies that document $n$ has higher discrimination score for cluster $j$ than all other clusters (including the current cluster $i$). Guided by our new discrimination score calculations, we assign the document $n$ to cluster $j$. This assignment results in a decrement of $\hat{d}_{ni}$ in discrimination score of cluster $i$ but an increment of $\hat{d}_{nj}$ in discrimination score of cluster $j$.

Summing the discrimination scores of all clusters will now produce new objective function value $J_{t+1}$ and

$$J_{t+1} > J_t$$

(3.16)

because of condition 3.15.

The migration of document $n$ from cluster $i$ to cluster $j$ produces three possible types of terms:

1) Terms of document $n$ whose weights increase in cluster $j$; let this subset of terms be $S_j$.
2) Terms of document $n$ whose weights decrease in cluster $j$; let this subset of terms be $S_i$.
3) Terms in document $n$ with unchanged weights.

But we see that

$$\sum_{a \in S_j} w_{aj} > \sum_{a \in S_i} w_{ai}$$

(3.17)

because of Eq. 3.16.

Therefore, there can be terms with decreased weights but overall to increase $J_{t+1}$ the total contribution of term weights is monotonically increased. And, for next iteration $J_{t+2} \geq J_{t+1}$.

Since $J$ is bounded above by the sum of discrimination scores of best possible clustering, CDIM will converge, although it may converge to some local maxima.

3.3 Variations of CDIM

In this section we present some variations of CDIM that we experiment.

3.3.1 CDIM-RR and CDIM-MDI

Two versions of CDIM are CDIM-RR and CDIM-MDI that are defined by using the discrimination measures relative risk (RR) and measurement of discrimination information (MDI) for calculation
of term weights respectively, identified in section 3.2.3. Document discrimination information is then calculated using equation 3.11. The CDIM algorithm steps are then iterated to optimize objective function of equation 3.1.

CDIM-MDI-S is the symmetric version of CDIM-MDI which is obtained by assigning equal prior probabilities while calculating the measurement of discrimination information, as described in section 3.2.3.

3.3.2 CDIM by Repeated Bisection

The original CDIM algorithm works in a $k$-way fashion, also called direct clustering. The data is initially partitioned into $k$ clusters and then these partitioned are refined iteratively to reach an optimum value of objective function. Another possible method is to split the data into two clusters and then keep on splitting a candidate cluster until the desired number of clusters $k$ is obtained. This type of algorithm is known as repeated bisection (RB) in literature [Steinbach and Karypis (2000)].

In repeated bisection version of CDIM, we select the largest cluster at each step and split it into two clusters using CDIM algorithm. Selecting largest cluster as bisection candidate is simple and using other complex selection methods do not give significantly better results in general [Steinbach and Karypis (2000), Zhao et al. (2005)]. Repeated bisection results are presented for both discrimination measures i.e. relative risk and MDI, yielding the versions CDIM-RR-RB and CDIM-MDI-RB respectively.

3.3.3 CDIM with Information Retrieval Measures DR and DC

Besides using discrimination measures in our clustering method, we experiment with relevance measures too. We combine Domain Relevance (DR) and Domain Consensus (DC) as proposed by [Navigli and Velardi (2004)], to assign cluster relevance weights to terms.

In our clustering context, domain relevance of a term $x_j$ to a cluster $C_k$ is calculated as

$$DR_{x_j,k} = \frac{p(x_j|C_k)}{\max_{1 \leq v \leq n} p(x_j|C_v)} \tag{3.18}$$
Domain consensus is computed to see the distributed use of a term in a cluster. A term with wide usage in a cluster i.e. present in many documents, is more relevant to the cluster than a term that occurs a large number of times in small number of documents in the cluster. Domain consensus for a term $x_j$ present in documents $i \in \mathcal{C}_k$ is computed as

$$DC_{x_j,k} = \sum_{i \in \mathcal{C}_k} p(x_j|i) \log \frac{1}{p(x_j|i)}$$

The two measures are then combined to calculate term weights

$$w_{jk} = \alpha DR_{x_j,k} + \beta DC_{x_j,k}^{\text{norm}}$$

The values for $\alpha$ and $\beta$ are chosen to be 0.9 and 0.3 respectively as recommended by [Navigli and Velardi (2004)]. DC is normalized by the number of documents in the cluster.

These term weights $w_{jk}$ are pooled to get document weights $d_{ik}$. The objective function for CDIM now maximizes the domain relevance and domain consensus as

$$J = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} r_{ik} d_{ik}$$

where $r_{ik} = 1$ if document $i$ is assigned to cluster $k$ and zero otherwise. We name this version of CDIM that uses domain relevance and domain consensus as CDIM-DRDC.

### 3.4 Experimental Setup

We conduct extensive experimental evaluations of our document cluster method. Our evaluations comprise of three sets of experiments. First, we evaluate the clustering quality of CDIM and compare it with other clustering methods on 10 text data sets. Second, we illustrate the understanding that is provided by CDIM clustering. Third, we evaluate certain implementation issues including term selection, convergence, and probability estimation. The results of these experiments are given in the next section. Here, we describe our experimental setup.
3.4.1 Data Sets

Our experiments are conducted on 10 standard text data sets of different sizes, contexts, and complexities. The key characteristics of these data sets are given in Table 3.1. Data set 1 is obtained from the Internet Content Filtering Group’s web site\(^1\), data set 2 is available from a Cornell University web page\(^2\), and data sets 3 to 10 are obtained from Karypis Lab, University of Minnesota\(^3\). Data sets 1 (stopword removal) and 3 to 10 (stopword removal and stemming) are available in preprocessed formats, while we perform stopword removal and stemming of data set 2.

We provide brief descriptions of each data set below.

The pu data set contain e-mails received by a particular user labeled as either spam or non-spam. The movie data set contain reviews of movies from the Internet Movie Database (IMDB). Each document is labeled as either a positive or a negative review. The hitech data set has newspaper articles belonging to one of six categories: computers, electronics, health, medical, research, and technology. The reviews data set contain articles about food, movies, music, radio, and restaurants. Both of these two data sets are derived from the San Jose Mercury newspaper articles that are distributed as part of the TREC (Text Retrieval Conference) collection (TIPSTER Vol. 3).

The seven and ten category data sets tr31 and tr41 are derived from TREC-6 and TREC-7 collections\(^4\). The categories in these data sets correspond to the queries and the documents that are judged most relevant to them.

The data sets re0 and re1 are taken from Reuters-21578 text categorization test collection distribution 1.0. The labels are divided into two sets producing two different data sets. The documents having single label only are used for both data sets.

The ohscal data set is derived from the OHSUMED collection of medical publications. It contain documents from ten categories: antibodies, carcinoma, DNA, in-vitro, molecular sequence data, pregnancy, prognosis, receptors, risk factors, and tomography. The wap data set is obtained from the WebACE project and each document corresponds to a web page listed in the subject hierarchy of Yahoo!

\(^1\)http://labs-repos.iit.demokritos.gr/skel/i-config/downloads/
\(^2\)http://www.cs.cornell.edu/People/pabo/movie-review-data/
\(^3\)http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download
\(^4\)http://trec.nist.gov
### Table 3.1: Data sets and their characteristics

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Documents (N)</th>
<th>Terms (M)</th>
<th>Categories (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pu</td>
<td>672</td>
<td>19868</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>movie</td>
<td>1200</td>
<td>38408</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>reviews</td>
<td>4069</td>
<td>23220</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>hitech</td>
<td>2301</td>
<td>13170</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>tr31</td>
<td>927</td>
<td>10128</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>tr41</td>
<td>878</td>
<td>7454</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>ohscal</td>
<td>11162</td>
<td>11465</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>re0</td>
<td>1504</td>
<td>2886</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>wap</td>
<td>1560</td>
<td>8460</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>re1</td>
<td>1657</td>
<td>3758</td>
<td>25</td>
</tr>
</tbody>
</table>

#### 3.4.2 Comparison Methods

We compare CDIM with five clustering methods. Four of them are $K$-means variants and one of them is based on Non-Negative Matrix Factorization (NMF) [Xu et al. (2003)].

The four $K$-means variants are selected from the CLUTO Toolkit [Karypis (2002)] based on their strong performances reported in the literature [Steinbach and Karypis (2000), Tan et al. (2006)]. Two of them are direct $K$-way clustering methods while the remaining two are repeated bisection methods that obtain $K$ clusters by repeatedly performing two-way partitioning of clusters. For each of these two types of methods, we consider two different objective functions. One objective function maximizes the sum of similarities between documents and their cluster mean. The direct and repeated bisection methods that use this objective function are identified as Direct-I2 and RB-I2, respectively. The second objective function that we consider maximizes the ratio of I2 and E1, where I2 is the intrinsic (based on cluster cohesion) objective function defined above and E1 is an extrinsic (based on separation) function that minimizes the sum of the normalized pairwise similarities of documents within clusters with the rest of the documents. The direct and repeated bisection methods that use this hybrid objective function are identified as Direct-H2 and RB-H2, respectively.

For NMF, we use the implementation provided in the DTU:Toolbox$^5$. Specifically, we use the multiplicative update rule with Euclidean measure for approximating the term-document matrix.

In using the four $K$-means variants, the term-document matrix is defined by term-frequency-inverse-document-frequency (TF-IDF) values and the cosine similarity measure is adopted for document comparisons. For NMF, the term-document matrix is defined by term frequency values.

---

Each method is run 10 times, every time starting with a random initialization of the clusters, and results are reported as average ± standard deviation of the performance measures.

**Reason of Selecting these Comparison Methods**

CDIM is an iterative partitional clustering algorithm. In each iteration, two steps are repeated: i) assignment of documents to clusters ii) update of term weights. This two step iterative nature of the algorithm makes CDIM highly suitable for comparison with different K-means variants that also work on similar two step iterative partitioning technique. Similarly, NMF is a popular method of text clustering and its relationship with K-means has also been established [Ding et al. (2005), Li and Ding (2006)]. However comparison with other approaches and testing on other datasets is a worthy future work.

### 3.4.3 Clustering Validation Measures

The validation measure that we use is the BCubed metric [Bagga and Baldwin (1998)]. In [Amigó et al. (2009)], Amigó et al. evaluate several extrinsic clustering validation measures both empirically and theoretically. They find that the BCubed precision and recall are the only measures that satisfy all desirable constraints for a good measure for clustering validation.

BCubed precision and recall are computed for each data object or document. Let $L(o)$ and $C(o)$ be the category and cluster of an object $o$. Then, the correctness of the relation between objects $o$ and $o'$ in the clustering can be defined as

$$Correct(o, o') = \begin{cases} 1 & \text{iff } L(o) = L(o') \leftrightarrow C(o) = C(o') \\ 0 & \text{otherwise} \end{cases}.$$  

This means that the relationship between two objects that share a category is correct if and only if they are in the same cluster. BCubed precision ($BP$) and BCubed recall ($BR$) can now be defined as follows:

$$BP = \text{Avg}_o [\text{Avg}_{o'.C(o')=C(o)} [Correct(o, o')]]$$

$$BR = \text{Avg}_o [\text{Avg}_{o'.L(o')=L(o)} [Correct(o, o')]]$$

48
To obtain a single evaluation value, BCubed precision and recall are combined using the harmonic mean formula:

$$BF = 2 \times \frac{BP \times BR}{BP + BR}$$

The BCubed F-measure ($BF$) ranges from 0 to 1 with larger values signifying better clusterings.

### 3.5 Results and Discussion

In this section, we present the results of our experimental evaluations. These are divided into clustering quality comparison, cluster understanding and visualization, some additional experiments, and discussion.

#### 3.5.1 Clustering Quality

A common approach to evaluating clustering methods is through clustering quality analysis. We compare the clusterings produced by CDIM to that produced by five popular methods on ten text data sets. The desired number of clusters $K$ for each data set is set equal to the number of categories in that data set (see Table 3.1). We report performance with average entropy and BCubed F-measure, averaged over 10 generated clusterings starting with random initial partitions.

Tables 3.3 and 3.4 compare the clustering quality of CDIM using the relative risk (RR) and the measurement of discrimination information (MDI) measures respectively. In each table, the highest average performance for each data set is highlighted in bold.

CDIM using relative risk outperforms its five competitors in six out of ten datasets, table 3.3. The pattern of consistently better performance for smaller values of $k$ is also visible. This is attributable to the lesser resolution power of the multi-way comparisons ($\hat{d}_{ik} = d_{ik} - \bar{d}_{ik}, \forall k$) that are required in CDIM-RR for document assignment.

The performance of CDIM-MDI is also very robust. Besides remaining unbeaten in seven out of ten datasets, its performance enhances with larger values of $k$. The incorporation of the log factor with priors based normalization, as in equations 3.4 and 3.5, causes this improvement.

Investigating the details of the datasets gives a better picture of the winning conditions for the
algorithms. Basic data statistics: average terms per document, average terms per category and average documents per category for each dataset are reported in Table 3.2. It is notable that for datasets having larger values of average number of terms per document e.g. values greater than ten, CDIM-RR outperforms others whereas for datasets having smaller values of average number of terms per document e.g. values less than or equal to six, CDIM-MDI outperforms others. The datasets where some other algorithm wins i.e. ohscal and tr41, the winning margin is not very high and the winner has score value within one standard deviation of CDIM. This observation about results based on number of average terms per document is shown in figure 3.1. Hence we find that for large documents, the simpler measure i.e. relative risk proves to be more powerful discrimination measure as compared to MDI. If documents in the dataset are short than MDI becomes more discriminative due to the use of prior probabilities and log factor calculations.

Table 3.5 presents the comparison of CDIM variants with each other. The columns give BCubed values of CDIM using relative risk, MDI, repeated bisection using relative risk and MDI, and MDI using symmetric priors, in order. The comparison proves that using MDI discrimination measure with CDIM produces more robust and scalable clustering.

Statistical Significance Tests

In order to validate our results we apply Friedman’s test, a non-parametric test popularly used for significance testing of multiple algorithms on multiple datasets Demsar (2006). Scores of both CDIM-RR and CDIM-MDI are found to be significantly better than Direct-I2, RB-I2 and NMF at 0.05 level of significance. Whereas the scores of CDIM-RR and CDIM-MDI both are not significantly different from Direct-I2 and RB-I2 at 0.05 level of significance.

Table 3.2: Data sets statistics (rounded off values). Terms (M), Documents (N), Categories (K).

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>M/N</th>
<th>M/K</th>
<th>N/K</th>
<th>Winner</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pu</td>
<td>30</td>
<td>9934</td>
<td>336</td>
<td>CDIM-RR</td>
</tr>
<tr>
<td>2</td>
<td>movie</td>
<td>32</td>
<td>19204</td>
<td>600</td>
<td>CDIM-RR</td>
</tr>
<tr>
<td>3</td>
<td>reviews</td>
<td>6</td>
<td>4644</td>
<td>814</td>
<td>CDIM-MDI</td>
</tr>
<tr>
<td>4</td>
<td>hitech</td>
<td>6</td>
<td>2195</td>
<td>384</td>
<td>CDIM-MDI</td>
</tr>
<tr>
<td>5</td>
<td>tr31</td>
<td>11</td>
<td>1447</td>
<td>132</td>
<td>CDIM-RR</td>
</tr>
<tr>
<td>6</td>
<td>tr41</td>
<td>8</td>
<td>745</td>
<td>88</td>
<td>Direct-I2</td>
</tr>
<tr>
<td>7</td>
<td>ohscal</td>
<td>1</td>
<td>1147</td>
<td>1116</td>
<td>RB-I2</td>
</tr>
<tr>
<td>8</td>
<td>re0</td>
<td>2</td>
<td>222</td>
<td>116</td>
<td>CDIM-MDI</td>
</tr>
<tr>
<td>9</td>
<td>wap</td>
<td>5</td>
<td>423</td>
<td>78</td>
<td>CDIM-MDI</td>
</tr>
<tr>
<td>10</td>
<td>rel</td>
<td>2</td>
<td>150</td>
<td>66</td>
<td>CDIM-MDI</td>
</tr>
</tbody>
</table>
Table 3.3: Comparison of clustering quality of CDIM-RR via the BCubed F-measure. Averages and standard deviations obtained from 10 runs are shown.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CDIM-RR</th>
<th>Direct-I2</th>
<th>Direct-H2</th>
<th>RB-I2</th>
<th>RB-H2</th>
<th>NMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>pu</td>
<td>0.732±0.08</td>
<td>0.565±0.02</td>
<td>0.553±0.02</td>
<td>0.565±0.02</td>
<td>0.553±0.02</td>
<td>0.612±0.04</td>
</tr>
<tr>
<td>movie</td>
<td>0.556±0.02</td>
<td>0.533±0.02</td>
<td>0.522±0.01</td>
<td>0.533±0.02</td>
<td>0.522±0.01</td>
<td>0.510±0.01</td>
</tr>
<tr>
<td>reviews</td>
<td>0.675±0.07</td>
<td>0.627±0.06</td>
<td>0.626±0.06</td>
<td>0.609±0.04</td>
<td>0.669±0.03</td>
<td>0.552±0.03</td>
</tr>
<tr>
<td>hitech</td>
<td>0.442±0.03</td>
<td>0.391±0.02</td>
<td>0.380±0.02</td>
<td>0.394±0.02</td>
<td>0.390±0.03</td>
<td>0.399±0.02</td>
</tr>
<tr>
<td>tr31</td>
<td>0.613±0.09</td>
<td>0.585±0.05</td>
<td>0.575±0.05</td>
<td>0.553±0.07</td>
<td>0.572±0.05</td>
<td>0.362±0.03</td>
</tr>
<tr>
<td>tr41</td>
<td>0.578±0.06</td>
<td><strong>0.608±0.02</strong></td>
<td>0.584±0.03</td>
<td>0.602±0.05</td>
<td>0.590±0.04</td>
<td>0.361±0.04</td>
</tr>
<tr>
<td>ohscal</td>
<td>0.428±0.04</td>
<td>0.422±0.02</td>
<td>0.417±0.03</td>
<td><strong>0.432±0.01</strong></td>
<td>0.427±0.01</td>
<td>0.250±0.02</td>
</tr>
<tr>
<td>re0</td>
<td>0.411±0.02</td>
<td>0.382±0.02</td>
<td>0.382±0.01</td>
<td>0.397±0.03</td>
<td>0.375±0.01</td>
<td>0.345±0.02</td>
</tr>
<tr>
<td>wap</td>
<td>0.445±0.02</td>
<td>0.462±0.01</td>
<td>0.444±0.01</td>
<td><strong>0.465±0.02</strong></td>
<td>0.438±0.02</td>
<td>0.299±0.02</td>
</tr>
<tr>
<td>rel</td>
<td>0.393±0.04</td>
<td><strong>0.443±0.02</strong></td>
<td>0.436±0.02</td>
<td>0.416±0.01</td>
<td>0.418±0.03</td>
<td>0.301±0.03</td>
</tr>
</tbody>
</table>

Table 3.4: Comparison of clustering quality of CDIM-MDI via the BCubed F-measure. Averages and standard deviations obtained from 10 runs are shown.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CDIM-MDI</th>
<th>Direct-I2</th>
<th>Direct-H2</th>
<th>RB-I2</th>
<th>RB-H2</th>
<th>NMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>pu</td>
<td>0.582±0.03</td>
<td>0.565±0.02</td>
<td>0.553±0.02</td>
<td>0.565±0.02</td>
<td>0.553±0.02</td>
<td><strong>0.612±0.04</strong></td>
</tr>
<tr>
<td>movie</td>
<td><strong>0.533±0.02</strong></td>
<td><strong>0.533±0.02</strong></td>
<td>0.522±0.01</td>
<td>0.533±0.02</td>
<td>0.522±0.01</td>
<td>0.510±0.01</td>
</tr>
<tr>
<td>reviews</td>
<td><strong>0.792±0.07</strong></td>
<td>0.627±0.06</td>
<td>0.626±0.06</td>
<td>0.609±0.04</td>
<td>0.669±0.03</td>
<td>0.552±0.03</td>
</tr>
<tr>
<td>hitech</td>
<td><strong>0.590±0.02</strong></td>
<td>0.391±0.02</td>
<td>0.380±0.02</td>
<td>0.394±0.02</td>
<td>0.390±0.03</td>
<td>0.399±0.02</td>
</tr>
<tr>
<td>tr31</td>
<td><strong>0.594±0.06</strong></td>
<td>0.585±0.05</td>
<td>0.575±0.05</td>
<td>0.553±0.07</td>
<td>0.572±0.05</td>
<td>0.362±0.03</td>
</tr>
<tr>
<td>tr41</td>
<td>0.508±0.03</td>
<td><strong>0.608±0.02</strong></td>
<td>0.584±0.03</td>
<td>0.602±0.05</td>
<td>0.590±0.04</td>
<td>0.361±0.04</td>
</tr>
<tr>
<td>ohscal</td>
<td>0.390±0.03</td>
<td>0.422±0.02</td>
<td>0.417±0.03</td>
<td><strong>0.432±0.01</strong></td>
<td>0.427±0.01</td>
<td>0.250±0.02</td>
</tr>
<tr>
<td>re0</td>
<td><strong>0.431±0.02</strong></td>
<td>0.382±0.02</td>
<td>0.382±0.01</td>
<td>0.397±0.03</td>
<td>0.375±0.01</td>
<td>0.345±0.02</td>
</tr>
<tr>
<td>wap</td>
<td><strong>0.478±0.03</strong></td>
<td>0.462±0.01</td>
<td>0.444±0.01</td>
<td>0.465±0.02</td>
<td>0.438±0.02</td>
<td>0.299±0.02</td>
</tr>
<tr>
<td>rel</td>
<td><strong>0.456±0.03</strong></td>
<td>0.443±0.02</td>
<td>0.436±0.02</td>
<td>0.416±0.01</td>
<td>0.418±0.03</td>
<td>0.301±0.03</td>
</tr>
</tbody>
</table>
Table 3.5: Comparison of clustering quality among CDIM variants via the BCubed F-measure. Averages and standard deviations obtained from 10 runs are shown.

<table>
<thead>
<tr>
<th>Data</th>
<th>CDIM-RR</th>
<th>CDIM-MDI</th>
<th>CDIM-RR-RB</th>
<th>CDIM-MDI-RB</th>
<th>CDIM-DRDC</th>
<th>CDIM-MDI-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>pu</td>
<td>0.732±0.08</td>
<td>0.582±0.03</td>
<td>0.732±0.08</td>
<td>0.582±0.03</td>
<td>0.687±0.08</td>
<td>0.582±0.03</td>
</tr>
<tr>
<td>movie</td>
<td>0.564±0.02</td>
<td>0.533±0.02</td>
<td>0.556±0.02</td>
<td>0.533±0.02</td>
<td>0.562±0.04</td>
<td>0.572±0.05</td>
</tr>
<tr>
<td>reviews</td>
<td>0.543±0.08</td>
<td>0.212±0.09</td>
<td>0.461±0.08</td>
<td>0.212±0.09</td>
<td>0.358±0.10</td>
<td>0.212±0.09</td>
</tr>
<tr>
<td>hitech</td>
<td>0.442±0.03</td>
<td>0.500±0.02</td>
<td>0.386±0.03</td>
<td>0.431±0.03</td>
<td>0.338±0.02</td>
<td>0.462±0.02</td>
</tr>
<tr>
<td>tr31</td>
<td>0.613±0.09</td>
<td>0.504±0.06</td>
<td>0.491±0.06</td>
<td>0.465±0.03</td>
<td>0.558±0.08</td>
<td>0.500±0.08</td>
</tr>
<tr>
<td>ohscal</td>
<td>0.428±0.04</td>
<td>0.399±0.03</td>
<td>0.375±0.02</td>
<td>0.283±0.02</td>
<td>0.351±0.02</td>
<td>0.318±0.02</td>
</tr>
<tr>
<td>re0</td>
<td>0.411±0.02</td>
<td>0.431±0.02</td>
<td>0.401±0.01</td>
<td>0.419±0.02</td>
<td>0.399±0.01</td>
<td>0.390±0.01</td>
</tr>
<tr>
<td>wap</td>
<td>0.445±0.02</td>
<td>0.478±0.03</td>
<td>0.395±0.03</td>
<td>0.363±0.02</td>
<td>0.396±0.02</td>
<td>0.427±0.02</td>
</tr>
<tr>
<td>rel</td>
<td>0.393±0.04</td>
<td>0.456±0.03</td>
<td>0.318±0.03</td>
<td>0.334±0.01</td>
<td>0.292±0.02</td>
<td>0.391±0.01</td>
</tr>
</tbody>
</table>

We tested also the significance of difference between the results of CDIM variants using Friedman’s test. CDIM-RR achieves top position with highest mean rank and significantly better performance than CDIM-RR-RB, CDIM-MDI-RB and CDIM-DRDC. CDIM-MDI gets second position with significantly better performance than CDIM-MDI-RB only. Notably, the performance difference of CDIM-RR and CDIM-MDI is not statistically significant.

### 3.5.2 Cluster Understanding and Visualization

A key application of data clustering is corpus understanding. In the case of document clustering, it is important that clustering methods output information that can readily be used to interpret the clusters and their documents. CDIM is based on term discrimination information and each of its cluster is describable by the highly discriminating terms in it.

We illustrate the understanding provided by CDIM’s output by displaying the top 10 most discriminating terms (stemmed words) for each cluster of the ohscal data set in Table 3.6. The ohscal data set contains publications from 10 different medical subject areas (antibodies, carcinoma, DNA, in-vitro, molecular sequence data, pregnancy, prognosis, receptors, risk factors, and tomography).

By looking at the top ten terms, it is easy to determine the category of most clusters: cluster 2 = carcinoma, cluster 3 = antibodies, cluster 4 = prognosis, cluster 5 = pregnancy, cluster 6 = risk factors, cluster 7 = DNA, cluster 9 = receptors, cluster 10 = tomography. The categories molecular sequence data and in-vitro do not appear to have a well-defined cluster; molecular sequence data has some overlap with cluster 7 while in-vitro has some overlap with clusters 1 and 9. Nonetheless, clusters 2 and 8 still give coherent meaning to the documents they contain.
As another example, in the hitech data set, the top 5 terms for two clusters are: (1) ‘health’, ‘care’, ‘patient’, ‘hospital’, ‘medical’, and (2) ‘city’, ‘council’, ‘project’, ‘build’, ‘water’. The first cluster can be mapped to the health category while the second cluster does not have an unambiguous mapping to a category but it still gives sufficient indication that these articles discuss hi-tech related development projects.

Since CDIM finds clusters in a $K$-dimensional discrimination information space, the distribution of documents among clusters can be visualized via simple scatter plots. The 2-dimensional scatter plot of documents in the pu data set is shown in Figure 3.2 for clustering produced by CDIM-RR. The $x$- and $y$-axes in this figure correspond to document discrimination information for cluster 1 and 2 ($d_{1i}$ and $d_{2i}$), respectively. When $d_{1i} > d_{2i}$ then document $i$ belongs to cluster 1, and vice versa. Thus, the two clusters are spread along the two axes starting from the origin. For illustration purposes, the true labels of the documents are also shown in Figure 3.2, indicating that a high quality clustering is obtained.

Such scatter plots can be viewed for any pair of clusters when $K > 2$. Since CDIM’s document assignment decision is based upon document discrimination scores ($\hat{d}_{ik}$, $\forall k$), scatter plots of documents in this space are also informative; each axis quantifies how relevant a document is to a cluster in comparison to the remaining clusters.

3.5.3 Additional Experiments and Discussion

We present some additional results and provide a discussion of some implementation issues in this section.

Table 3.6: Top 10 most discriminating terms (stemmed words) for clusters in ohscal data set

<table>
<thead>
<tr>
<th>$k$</th>
<th>Top 10 terms in cluster $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'platelet', 'kg', 'mg', 'dose', 'min', 'plasma', 'pressur', 'flow', 'microgram', 'antagonist'</td>
</tr>
<tr>
<td>2</td>
<td>'carcinoma', 'tumor', 'cancer', 'surviv', 'chemotherapi', 'stage', 'recurr', 'malign', 'resect', 'therapi'</td>
</tr>
<tr>
<td>3</td>
<td>'antibodi', 'antigen', 'virus', 'anti', 'infect', 'hiv', 'monoclon', 'lg', 'inmun', 'sera'</td>
</tr>
<tr>
<td>4</td>
<td>'patient', 'complic', 'surgeri', 'ventricular', 'infarct', 'eye', 'coronari', 'cardiac', 'morta'</td>
</tr>
<tr>
<td>5</td>
<td>'pregnan', 'fetal', 'gestat', 'matern', 'women', 'infant', 'deliveri', 'birth', 'labor', 'pregnant'</td>
</tr>
<tr>
<td>6</td>
<td>'risk', 'alcohol', 'age', 'children', 'cholesterol', 'health', 'factor', 'women', 'preval', 'popul'</td>
</tr>
<tr>
<td>7</td>
<td>'gene', 'sequence', 'dna', 'mutat', 'protein', 'chromosom', 'transcript', 'rna', 'amino', 'structur'</td>
</tr>
<tr>
<td>8</td>
<td>'contract', 'muscle', 'relax', 'micron', 'calcium', 'effect', 'respon', 'antagonist', 'releas', 'action'</td>
</tr>
<tr>
<td>9</td>
<td>'il', 'receptor', 'cell', 'stimul', 'bind', 'growth', 'gamma', 'alpha', 'insulin', '0'</td>
</tr>
<tr>
<td>10</td>
<td>'ct', 'imag', 'comput', 'tomographi', 'scan', 'lesion', 'magnet', 'reson', 'cerebr', 'tomograph'</td>
</tr>
</tbody>
</table>
Term Discrimination Information Distribution and Term Selection

Typically, the distribution of term discrimination information has a long narrow tail. In other words, a small fraction of terms provide high discriminative power while the vast majority provide little discriminative power. This observation suggests that term selection via the parameter $t$ can reduce the space complexity of CDIM without impacting clustering quality significantly. Except for the experiments reported in this subsection, all other results are obtained with $t = 0$.

We evaluate the impact of term selection by comparing clustering quality with increasing value of term selection parameter $t$. Figure 3.3 shows the variation of clustering quality using CDIM-RR with sum of significant terms for both clusters ($|T_1| + |T_2|$) in the pu data set. It is seen that clustering quality does not degrade much with significant decrease in number of terms. In fact, when the number of terms is reduced to 875 from 19,868 the clustering quality actually improves. These results show that CDIM is scalable and robust to term selection.

Convergence

In practice, smooth and rapid convergence of algorithms is highly desirable. CDIM maximizes its objective function by using a two-step greedy procedure that ensures that the objective function is non-decreasing from one iteration to the next. This procedure corresponds to the nave semi-
supervised learning approach of labeling and learning, which has also been shown to be convergent [Xue and Weiss (2009)].

Figure 3.4 shows the converge curves of CDIM on five data sets. This figure demonstrate the smooth convergence behavior of CDIM, and highlights the fact that convergence is achievable within 15 iterations.

CDIM is able to find a local optimum solution only, dependent on the initial partitioning chosen. This characteristic, however, is present in many partitional clustering methods like the \( K \)-means algorithm.

Figure 3.4: Convergence curves of CDIM
3.6 Conclusion

In this chapter, we present and evaluate a new document clustering method, CDIM. CDIM finds clusters in a $K$-dimensional space in which documents are well discriminated. It does this by maximizing the sum of the discrimination information provided by documents for their clusters minus that provided for the remaining clusters. Document discrimination information is computed from the discrimination information provided by the terms in it. Term discrimination information is estimated from the document collection via its relative risk. An advantage of using a measure of discrimination information is that it also quantifies the degree of relatedness of a term to its context in the collection. Thus, CDIM produces clusters that are readily interpretable by their highly discriminating terms.

We conduct extensive experimental evaluations of CDIM. We compare its cluster quality with that of five popular clustering methods on ten data sets. CDIM outperforms NMF-based clustering and its performance is on par with the best $K$-means variant. We also illustrate the understanding provided by CDIM’s output, enabling documents in the clusters to be identifiable as belonging to specific contexts or topics.
Chapter 4

Content-Based Tag Recommendation using Discriminative Clustering

After completing the theoretical aspects and evaluation of our discriminative clustering algorithm CDIM in previous chapter, a practical application is presented in this chapter. It is a content-based tag recommendation system for social bookmarking systems based on discriminative clustering. Clustering is used to group posts based on their textual contents.

4.1 Introduction

Social bookmarking systems have become popular in recent years for organizing and sharing resources on the Web. Such systems allow users to build a database of resources, typically Web pages and publications, by adding basic information (such as URLs and titles) about them and by assigning one or more keywords or tags describing them. The tags serve to organize the resources and help improve recall in searches. Individual users databases are shared among all users of the system enabling the development of an information repository which is commonly referred to as a folksonomy [Golder and Huberman (2005)]. A folksonomy is a collection of users, resources, and tags assigned by a user to a resource posted by him or her. Tag recommendation for new posts by users is desirable for two reasons. First, it ensures uniformity of tagging enabling better searches, and second, it eases the task of users in selecting the most descriptive keywords for tagging the
Tag recommendation can have one of two goals: (1) to suggest tags tailored to individual users’ preferences (the ‘local’ goal) and (2) to suggest tags that promote uniformity in tagging of resources (the ‘global’ goal). Tag recommendation can benefit from the tagging history of users and resources. However, when a user posts for the first time and/or the posted resource is new this historical information is less useful. In such cases, content-based tag recommendation is necessary, in which the contents of the resource are relied upon for tag recommendation.

In our tag recommendation approach, we cluster the historical data of posted resources and develop a ranked list of discriminating tags and content terms for each cluster. Term discrimination information is calculated using relative risk as discussed in chapter 3. Given a new posting, based on its contents, we find the best three clusters and develop a weighted list of tags and terms appropriate for tagging the post. If the posters tagging history is available, then this provides a third ranked list of tags appropriate for the post. The final tag recommendation for the post is done by rules that select terms from the weighted lists. These rules also decide on the number of tags to recommend for each known poster. Extensive performance results are presented for the Bibsonomy data [Hotho et al. (2009)]. The rest of the chapter is organized as follows. Section 2 presents details of our content-based tag recommendation approach. Data preprocessing and analysis is discussed in Section 3. The results of our approach are presented and discussed in Section 5. We conclude in Section 6.

4.2 Discriminative Clustering for Content Based Tag Recommendation

Our approach for content-based tag recommendation in social bookmarking systems is based on discriminative clustering, content terms and tags rankings, and rules for final recommendations. We use our novel and efficient discriminative clustering method to group posts based on the tags assigned to them and based on their contents’ terms. This method maximizes the sum of the discrimination information provided by posts and outputs a weighted list of discriminating tags.
and terms for each cluster. We also maintain a ranked list of tags for seen users. Tags are suggested from these three rankings by intuitive rules that fuse the information from the lists. The rest of this section presents our approach in detail.

4.2.1 Problem Definition and Notation

A social bookmarking system, such as BibSonomy [BibSonomy (2009)], allows users to post and tag two kind of resources: Web bookmarks and publications. Each resource type is described by a fixed set of textual fields. A bookmark is described by fields like URL, title, and description, while a publication is described by fields in the standard bibtex record. Some of these fields (like title for bookmarks) are mandatory while others are optional. This textual information forms the content of the resource. Each user who posts a resource must also assign one or more tags for describing the resource.

Let \( p_i = \{u_i, x_i, t_i\} \) denotes the \( i \)th post, where \( u_i \) is the unique user/poster ID, and \( x_i \) and \( t_i \) are the vector space representations of the post’s contents and tags, respectively. If \( T \) is the size of the vocabulary then the \( i \)th post’s contents and tags can be written as \( x_i = \{x_{i1}, x_{i2}, \ldots, x_{iT}\} \) and \( t_i = \{t_{i1}, t_{i2}, \ldots, t_{iT}\} \), respectively, where \( x_{ij} \) (\( t_{ij} \)) denotes the frequency of term \( j \) (tag \( j \)) in post \( i \). Note that an identical vector space model is used to represent both content terms and tags, \( t_{ij} \in \{0, 1\}, \forall i, j \), and \( x_{ij} \geq 0, \forall i, j \). The historical data contain \( N \) posts. The tag recommender suggests tags for a new post \( i \) described by \( u_i \) and \( x_i \). The user \( u_i \) and resource described by content \( x_i \) may or may not appear in the historical data.

Let \( TG(i), TM(i), \) and \( TU(i) \) be the ranked list of tags from clustering, terms from clustering, and user tags, respectively, corresponding to the \( i \)th post. The actual tags recommended for post \( i \), denoted by \( TR(i) \), are determined from these ranked lists by intuitive rules.

Given a test data containing \( M \) posts, the performance of the tag recommender is evaluated by averaging F1-score of each prediction over the entire test data.
4.2.2 Discriminative Clustering for Tag and Term Ranking

The historical data of $N$ posts is clustered into $K \ll N$ groups using a novel discriminative clustering method. This method is motivated from the recently proposed DTWC algorithm for text classification [Junejo and Karim (2008)]. It is an iterative partitioning method that maximizes the sum of discrimination information provided by each textual content (a post, in our setting) between its assigned cluster and the remaining clusters. The key ideas include discriminative term weighting, discrimination information pooling, and discriminative assignment. Unlike other partitioning clustering methods, this method does not require the explicit definition of a similarity measure and a cluster representative. Furthermore, it builds a ranked list of discriminating terms for each cluster implicitly. The method is computationally more efficient than popular methods like the k-means clustering algorithm. We perform two clusterings of the historical data – one based on the content terms $x$ and the other based on the tags $t$ of the posts in the data. In the following description, we develop the method for content terms only; the method as applied to tags will be similar.

First, an initial clustering of the data is done. This can be done randomly or, less efficiently especially for large collections, by a single iteration of the k-means algorithm with the cosine similarity measure. Given this clustering, a discriminative term weight $w^k_j$ is computed for each term $j$ in the vocabulary and for each cluster $k$ as [Junejo and Karim (2008)]

$$w^k_j = \begin{cases} 
p(x_j|k)/p(x_j|\neg k) & \text{when } p(x_j|k) > p(x_j|\neg k) \\
p(x_j|\neg k)/p(x_j|k) & \text{otherwise} \end{cases}$$

where $p(x_j|k)$ and $p(x_j|\neg k)$ are the probabilities that term $j$ belongs to cluster $k$ and the remaining clusters ($\neg k$), respectively. The discriminative term weight quantifies the discrimination information that term $j$ provides for cluster $k$ over the remaining clusters. Note that this weight is expressed as a probability ratio and is always greater than or equal to 1. The probabilities are computed by maximum likelihood estimation from the historical data.

Having computed the discriminative term weights for the current clustering, two discrimination scores can be computed for each post $i$. One score, denoted as $Score^k(x_i)$, expresses the
discrimination information provided by post $i$ for cluster $k$, whereas the other score, denoted as $Score^{-k}(x_i)$, expresses the discrimination information provided by post $i$ for clusters $\neg k$. These scores are computed by linearly pooling the discrimination information provided by each term $x_j$ in post $i$ as [Junejo and Karim (2008)]

$$Score^k(x_i) = \frac{\sum_{j \in Z^k} x_j w^k_j}{\sum_j x_j} \quad \text{and} \quad Score^{-k}(x_i) = \frac{\sum_{j \in Z^{-k}} x_j w^{-k}_j}{\sum_j x_j}$$

In these equations, $Z^k = \{j | p(x_j | k) > p(x_j | \neg k)\}$ and $Z^{-k} = \{j | p(x_j | \neg k) > p(x_j | k)\}$ are sets of term indices that vouch for clusters $k$ and $\neg k$, respectively. Each post, described by its contents $x$, is then reassigned to the cluster $k$ for which the cluster score $f^k = Score^k(x) - Score^{-k}(x)$ is maximum. This is the cluster that makes each post most discriminating among all the clusters.

The overall clustering objective is to maximize the sum of discrimination information, or cluster scores, of all posts. Mathematically, this is written as

$$\text{Maximize } J = \sum_{i=1}^{N} \sum_{k=1}^{K} I^k(x_i) \cdot f^k$$

where $I^k(x_i) = 1$ if post $i$ is assigned to cluster $k$ and zero otherwise. Iterative reassignment is continued until the change in the clustering objective becomes less than a specified small value. Typically, the method converges satisfactorily in fewer than 15 iterations.

The discriminative term weights for the terms in the index set $Z^k$ are ranked to obtain the weighted and ranked list of terms for cluster $k$. As mentioned earlier, clustering is also performed based on the tags assigned to posts. This clustering yields another weighted and ranked list of tags for each cluster.

It is worthwhile to point out that the term-based clustering is done on both the training and testing data sets. This approach allows the terms that exist only in the test data to be included in the vocabulary space, and for such terms to be available for recommendation as tags.

Given a new post $i$ described by $x_i$, the best cluster for it is the cluster $k$ for which the cluster
score $f^k$ is a maximum. The corresponding ranked list of terms and tags for post $i$ are denoted by $TM(i)$ and $TG(i)$, respectively. These ranked lists contain the most discriminating tags for post $i$ based on its contents.

### 4.2.3 Final Tag Recommendation

Given a new post, and based on the contents $x$ of the post, two ranked lists of terms appropriate for tagging are generated by the procedures described in the previous section. If the user of the post appears in the historical data, then an additional list of potential tags can be generated. This is the ranked list of tags $TU(i)$ used by the user of post $i$. The ranking is done based on frequency. Moreover, the average number of tags per user is computed and used while recommending tags for seen users.

The final list of tags for post $i$ is made by simple and intuitive rules that combine information from all the lists. Let $S$ be the number of tags to recommend for post $i$. Then, the final list of tags for the post is given by the following algorithm:

\[
TR(i) = TG(i)[1 : P] \cap TM(i)[1 : Q]
\]

\[
\text{IF } TU(i) \neq \emptyset \text{ THEN } TR(i) = TR(i) \cap TU(i)[1 : R]
\]

\[
\text{IF } |TR(i)| < S \text{ THEN add top terms from } TG(i), TM(i) \text{ in } TR(i)
\]

In the above algorithm, $P$, $Q$, and $R$ are integer parameters that define how many top terms to include from each list. If after taking the set intersections $|TR(i)| < S$ then the remaining tags are obtained from the top tags and terms in $TG(i)$ and $TM(i)$, respectively. In general, as seen from our evaluations, $R \leq Q \leq P$, indicating that $TG(i)$ is the least noisy source and $TU(i)$ the most noisy source for tags.
4.3 Evaluation Setup

4.3.1 Data and their Characteristics

We evaluate our approach on data sets made available by the ECML PKDD Discovery Challenge 2009 [Eisterlehner et al. (2009)]. These data sets are obtained from dumps of public bookmark and publication posts on BibSonomy [BibSonomy (2009)]. The dumps are cleaned by removing spammers’ posts and posts from the user dblp (a mirror of the DBLP Computer Science Bibliography). Furthermore, all characters from tags that are neither numbers nor letters are removed. UTF-8 encoding and unicode normalization to normal form KC are also performed.

The post-core at level 2 data is obtained from the cleaned dump (until 31 December 2008) and contain all posts whose user, resource, and tags appear in at least one more post in the post-core data. The post-core at level 2 contain 64,120 posts (41,268 bookmarks and 22,852 publications), 1,185 distinct users, and 13,276 distinct tags. We use the first 57,000 posts (in content ID order) for training and the remaining 7,120 posts for testing.

We also present results on the test data released as part of task 1 of the ECML PKDD Discovery Challenge 2009. This data is cleaned and processed as described above, but it contain only those posts whose user, resource, or tags do not appear in the post-core at level 2 data. This data contain 43,002 posts (16,898 bookmarks and 26,104 publications) and 1,591 distinct users. For this evaluation, we use the entire 64,120 posts in the post-core at level 2 for training and test on the 43,002 posts in the test data.

These data sets are available in the form of 3 tables – tas, bookmark, and bibtex – as described below. The content of a post is defined by the fields in the bookmark and bibtex tables, while the tags appear in the tas table.

**tas** fact table; who attached which tag to which post/content. Fields include: user (number; user names are anonymized), tag, content id (matches bookmark.content id or bibtex.content id), content type (1 = bookmark, 2 = bibtex), date

**bookmark** dimension table for bookmark data. Fields include: content id (matches tas.content id), url hash (the URL as md5 hash), url, description, extended description, date
bibtex  dimension table for BibTeX data. Fields include: content id (matches tas.content id), jour-
nal, volume, chapter, edition, month, day, booktitle, howPublished, institution, organization,
publisher, address, school, series, bibtexKey (the bibtex key (in the @... line)), url, type,
description, annotate, note, pages, bKey (the “key” field), number, crossref, misc, bibtexAb-
stract, simhash0 (hash for duplicate detection within a user – strict – (obsolete)), simhash1
(hash for duplicate detection among users – sloppy –), simhash2 (hash for duplicate detection
within a user – strict –), entrytype, title, author, editor, year

A few tagging statistics from the post-core data are given in Table 4.1 and Figure 4.1. These
statistics are used to fix the parameter $S$ (number of recommended tags) for known users. For
unseen users, $S$ is set at 5.

<table>
<thead>
<tr>
<th></th>
<th>Avg</th>
<th>Min</th>
<th>Max</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of tags per post</td>
<td>4</td>
<td>1</td>
<td>81</td>
<td>3.3</td>
</tr>
<tr>
<td>No. of posts per user</td>
<td>54</td>
<td>2</td>
<td>2031</td>
<td>162.9</td>
</tr>
<tr>
<td>No. of tags per user</td>
<td>62</td>
<td>1</td>
<td>4711</td>
<td>214.5</td>
</tr>
<tr>
<td>Frequency of tags</td>
<td>19</td>
<td>2</td>
<td>4474</td>
<td>106.9</td>
</tr>
</tbody>
</table>

Figure 4.1: Number of tags assigned to posts by users
4.3.2 Data Preparation

We explore tag recommendation performance on original contents, contents that have been augmented by crawled information, and contents that have been augmented and lemmatized.

The vocabulary for the vector space representation is formed from the tags and content terms in the training and testing sets. Selected content fields are used for gathering the content terms. For bookmark posts, the selected fields are url, description, and extended. For publication posts, the selected bibtex fields are booktitle, journal, howpublished, publisher, series, bibtexkey, url, description, annotate, note, bkey, crossref, misc, bibtexAbstract, entrytype, title, and author. As mentioned earlier, the tags, which appear in the tas table, are also included in the vocabulary.

We remove all the non-letter and non-digit characters, but retain umlauts and other non-Latin characters due to UTF-8 encoding. All processed terms of length greater than or equal to three are retained. The tags are processed similarly, but without considering the token length constraint.

Crawling

Crawling is done to fill in and augment important fields. For bookmark posts, the extended description field is appended with textual information from <TITLE>, <H1> and <H2> HTML fields of the URL provided in the posts.

For publication posts, missing abstract field are filled using online search. We use the publication title to search for its abstract on CiteULike [CiteULike (2009)]. If the article is found, and its abstract is available on CiteULike, the bibtexAbstract field of the post is updated. CiteULike is selected because its structure is simpler and it does not have any restrictions on the number of queries (in a day for example).

Lemmatization

We also explore lemmatization of the vocabulary while developing the vector space representation. Lemmatization is different from stemming as lemmatization returns the base form of a word rather than truncating it. We do lemmatization using TreeTagger [TreeTagger (2009)]. TreeTagger is capable of handling multiple languages besides English. We lemmatize the vocabulary using English,
French, German, Italian, Spanish and Dutch languages. The procedure, in brief, is as below:

1. TreeTagger is run on the vocabulary file once for each language: English, French, German, Italian, Spanish and Dutch.

2. TreeTagger returns the output file containing token, pos, lemma. The lemma is “<unknown>” if a token is not recognized in that language.

3. Using this “<unknown>” word, we combine the output of all six lemmatized files. If a term is not recognized by any language, the term itself is used as lemma.

4. If a word is lemmatized by more than one language, thenlemmas are prioritized in the sequence: English, French, German, Italian, Spanish, Dutch. The first lemma for the word is selected.

4.3.3 Evaluation Criteria

The performance of tag recommendation systems is typically evaluated using precision, recall, and F1 score, where the F1 score is a single value obtained by combining both precision and recall. We report the precision, recall, and F1 score averaged over all the posts in the testing set.

4.4 Results

In this section, we present and discuss the results of our discriminative clustering approach for content based tag recommendation. We start off by evaluating the performance of the clustering method.

4.4.1 Clustering Performance

The performance of the discriminative clustering method is evaluated on the entire 64,120 posts of the post-core at level 2 data. We cluster these posts based on the tags assigned to them. After clustering and ranking of tags for each cluster, we recommend the top 5 tags from the ranked list for all posts in each cluster. The average precision, recall, and F1 score percentages obtained for different values of $K$ (number of desired clusters) is shown in Table 4.2.
The top 5 tags become increasingly accurate recommendations as the number of clusters is increased, with the maximum recall of 48.7% and F1 score of 30.6% obtained when $K = 300$. These results simulate the scenario when the entire tag space (containing 13,276 tags) is known. Furthermore, there is no separation between training and testing data. Nonetheless, the results do highlight the worth of clustering in grouping related posts that can be tagged similarly.

Table 4.3 shows the top ranked tags for selected clusters. It is seen that the discriminative clustering method is capable of grouping posts and identifying descriptive tags for each group of posts. Noisy tags are not ranked high in the lists. It is even able to discriminate and group posts of different languages (not shown in this table), especially when clustering is based on content terms. Two valuable characteristics of the discriminative clustering method are its stability and efficiency. The method converges smoothly (Figure 4.2) usually within 15 iteration. More importantly, especially considering the large post by vocabulary sizes involved, is the efficiency of the method. Each iteration of the method completes within 3 minutes, even for the large 107,122 × 317,283 data for the content-based clustering of the post-core plus task 1 test data of ECML discovery challenge 2009.

### 4.4.2 Tag Recommendation Using $TG$ and $TM$ Only

In this section, we discuss the performance of recommending the top 5 tags from the $TG(i)$ or $TM(i)$ list of each post $i$. This evaluation is done on the testing data of 7,120 posts held out from the post-core at level 2 data. The clustering model is based on the first 57,000 posts (in content ID order) from the data. In this evaluation, the original data, without augmentation with crawled information, is used for creating the vector space representation.

Table 4.2: Performance of discriminative clustering of posts using the tags assigned to them (post-core at level 2 data)

<table>
<thead>
<tr>
<th>$K$</th>
<th>10</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td>Act. Clusters</td>
<td>10</td>
<td>48</td>
<td>95</td>
<td>189</td>
<td>274</td>
</tr>
<tr>
<td>Av. Precision (%)</td>
<td>12.5</td>
<td>19.2</td>
<td>22.3</td>
<td>25.2</td>
<td>26.9</td>
</tr>
<tr>
<td>Av. Recall (%)</td>
<td>21.0</td>
<td>32.8</td>
<td>38.6</td>
<td>45.9</td>
<td>48.7</td>
</tr>
<tr>
<td>Av. F1-score (%)</td>
<td>13.7</td>
<td>21.4</td>
<td>25.0</td>
<td>28.7</td>
<td>30.6</td>
</tr>
</tbody>
</table>
Table 4.3: Top tags for selected clusters (K = 200)

<table>
<thead>
<tr>
<th>No.</th>
<th>Top Discriminating Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>svm, ki2007webmining, mining, kernels, textmining, dm, textclassification</td>
</tr>
<tr>
<td>2</td>
<td>windows, freeware, utility, download, utilities, win, shareware</td>
</tr>
<tr>
<td>3</td>
<td>fun, flash, games, game, microfiction, flashfiction, sudden</td>
</tr>
<tr>
<td>4</td>
<td>tag, cloud, tagcloud, tags, folksonomia, tagging, vortragmunchen2008</td>
</tr>
<tr>
<td>5</td>
<td>library, books, archive, bibliothek, catalog, digital, opac</td>
</tr>
<tr>
<td>6</td>
<td>voip, mobile, skype, phone, im, messaging, hones</td>
</tr>
<tr>
<td>7</td>
<td>rss, feeds, aggregator, feed, atom, syndication, opml</td>
</tr>
<tr>
<td>8</td>
<td>bookmarks, bookmark, tags, bookmarking, delicious, diigo, socialbookmarking</td>
</tr>
</tbody>
</table>

Figure 4.2: Discriminative clustering convergence curves (clustering posts based on tags)

The recommendation results for different K values are given in Table 4.4. Results are shown for the case when only the top cluster for each post is considered, and for the case when the top three clusters of each post are merged in a weighted manner (using cluster score and discriminative term weights). It is observed that merging the lists of the top three clusters always gives better performance. Moreover, recommendations based on $T_{G}(i)$ are always better than those based on $T_{M}(i)$ indicating that the term-based clustering is more noisy than that based on tags. We also find out that $K = 200$ yields the highest recommendation performances.

4.4.3 Tag Recommendation Using All Lists

In this section, we evaluate the performance of our approach when utilizing information from all lists. We also evaluate performance on original, crawled, and crawled plus lemmatized data. These results are shown in Table 4.5. For this evaluation, we fix $K = 200$ and use the top three clusters.
Table 4.4: Tag recommendation performance (average F1-score percentages) using $TG$ or $TM$ only for original data.

<table>
<thead>
<tr>
<th>K</th>
<th>10</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TG$ Only (Best Cluster)</td>
<td>6.6</td>
<td>7.4</td>
<td>8.7</td>
<td>8.7</td>
<td>7.2</td>
</tr>
<tr>
<td>$TG$ Only (Top 3 Clusters)</td>
<td>7.3</td>
<td>8.2</td>
<td>9.5</td>
<td>10.6</td>
<td>9.1</td>
</tr>
<tr>
<td>$TM$ Only (Best Cluster)</td>
<td></td>
<td></td>
<td></td>
<td>6.3</td>
<td></td>
</tr>
<tr>
<td>$TM$ Only (Top 3 Clusters)</td>
<td></td>
<td></td>
<td></td>
<td>7.8</td>
<td></td>
</tr>
</tbody>
</table>

for building $TG(i)$ and $TM(i)$.

The first column (identified by the heading $TF$) shows the baseline result of recommending the top 5 most frequent tags in the training data (57,000 posts from post-core data). It is seen that our clustering based recommendation improves performance beyond the baseline performance. The second and third columns show the performance of recommending the top 5 terms from $TG(i)$ and $TM(i)$, respectively. The predictions of the tag-based clustering always outperform the predictions of the term-based clustering. In the fourth column, we report results for the case when the top 5 recommended tags are obtained by combining $TG(i)$ and $TM(i)$, as described in Section 3.3. These results are significantly better than those produced by each list independently.

The fifth column shows the results of combining all lists, including the user list $TU(i)$ when known. This strategy produces the best F1 score of 15.5% for the crawled data. This is a significant improvement over the baseline F1 score of 7.0%.

Table 4.5 also shows that filling in missing fields and augmenting the fields with crawled information improves performance. Lemmatization does not help, probably because users do not necessarily assign base forms of words as tags.

Table 4.5: Tag recommendation performance (average F1-score percentages) for processed data (K = 200; prediction based on top 3 clusters). The bottom line shows performance on ECML-DC09 task 1 test data.

<table>
<thead>
<tr>
<th>Data / Lists</th>
<th>$TF$</th>
<th>$TG$</th>
<th>$TM$</th>
<th>$TG, TM$</th>
<th>$TG, TM, TU$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Contents</td>
<td>7.0</td>
<td>10.6</td>
<td>7.8</td>
<td>11.5</td>
<td>12.8</td>
</tr>
<tr>
<td>Crawled Contents</td>
<td>7.0</td>
<td>12.3</td>
<td>10.4</td>
<td>14.3</td>
<td>15.5</td>
</tr>
<tr>
<td>Crawled+Lemmatized Contents</td>
<td>7.0</td>
<td>11.7</td>
<td>9.7</td>
<td>13.3</td>
<td>14.6</td>
</tr>
<tr>
<td>Task 1 Test Data (Crawled)</td>
<td>1.1</td>
<td>4.9</td>
<td>3.2</td>
<td>5.2</td>
<td>5.4</td>
</tr>
</tbody>
</table>
4.4.4 Tag Recommendation for Task 1 Test Data of ECML DC 2009

We report the performance of our approach on task 1 test data released by the ECML discovery challenge organizers [Eisterlehner et al. (2009)] on the bottom line of Table 4.5. We filled in missing and augmented other fields by crawled information. No lemmatization is done. The final vocabulary size is equal to 317,283 terms making the tag recommendation problem very sparse. The baseline performance of using the 5 most frequent tags from the post-core at level 2 (the training data for this evaluation) is the F1 score of 1.1% only. By using our discriminative clustering approach, the average F1 score reaches up to 5.4%. This low value is attributable to the sparseness of the data, and it is unlikely that other methods can cope better without extensive semantic normalization and micro modeling of the tagging process. The best score for challenge was 18.7% [Lipczak et al. (2009)]. One reason for our low performance was use of post-core level 2 data instead of using full data.

4.5 Conclusion

In this chapter, we explore a discriminative clustering approach for content-based tag recommendation in social bookmarking systems. We perform two clusterings of the posts: one based on the tags assigned to the posts and the second based on the content terms of the posts. The clustering method produces ranked lists of tags and terms for each cluster. The final recommendation is done by using both lists, together with the user’s tagging history if available. Our approach produces significantly better recommendations than the baseline recommendation of most frequent tags.
Chapter 5

Self-Optimizing, Self-Calibrating and Personalized Tag Recommendation

Self-optimization is one of the four basic self-* properties of autonomic systems. The others are self-configuration, self-healing and self-protection. Autonomic systems are the systems that manage themselves. Self-optimization is a desired property for large and complex software systems in order to achieve self-management and replace slow and expensive human hands.

We propose a new property and call it self-calibration. Self-calibration is an enabling property that helps a system achieve other properties e.g. self-optimization with minimal system changes. This leads to a system that becomes autonomic, also named as adaptive system in literature. Self-calibration helps systems achieve self-optimization and self-healing capabilities by executing smaller and targeted error correction or performance enhancing operations.

Personalization helps in developing systems that fulfil their goals at person to person level. All these important characteristics are discussed in this chapter in the context of our clustering based tag recommendation system established in previous chapter.
5.1 Introduction

5.1.1 Self-Optimization

Automatic tag recommendation systems are typically built once and then used for a long period of time. However, the recommendation performance of such systems degrades with time as the social environment evolves but the tag recommendation system does not. It is desirable for a tag recommendation system to be self-optimizing whereby it remains updated with fresh knowledge and is capable of accurate recommendations over time. Based on monitoring of recommendation performance, and given the administrator’s specification of average update time, the system should adapt automatically using optimal parameters.

In this chapter, we present and evaluate a self-optimizing strategy for our clustering based tag recommendation system for social bookmarking applications. We adopt the discriminative clustering based tag recommender presented in previous chapter. In this approach, the historical data of posted resources is clustered and a ranked list of discriminating tags for each cluster is developed. Given a new posting, based on its contents, the approach recommends the top 5 tags from the cluster that is most relevant to the post.

Our self-optimization strategy is empirical in nature. We analyze the performance of our tag recommendation system under different parameter settings. We observe that the performance of the system degrades with time, which helps us to decide when to update the system. We also develop a nonlinear optimization model for selecting the optimal parameters for maximum recommendation accuracy given constraint on the update time. The relationships in the optimization model are determined empirically by curve fitting. The solution of the optimization model tells us how best to update the recommendation system given administrator constraint on time. Our self-optimizing tag recommendation system is evaluated on real social bookmarking system data of Bibsonomy [BibSonomy (2009)] provided by ECML PKDD Discovery Challenge 2009 [Eisterlehner et al. (2009)]. Our experiments demonstrate that the self-optimizing strategy can improve (and in general maintain) the performance of the tag recommendation system with minimal intervention from the administrator.
5.1.2 Self-Calibration

A range of time and content driven autonomous intelligent systems such as spam filtering software, forecasting engines, online tag recommendation, etc. are based on off-line learning algorithms. Here by content driven engine we mean systems where operations, goals or primary services are dependent on and driven by content. In these systems the costly step of model creation is done off-line and decisions are made during execution using this model [Hassan et al. (2009b), Javed and Arshad (2009), Junejo and Karim (2007)]. Since the content changes over time, the static off-line model becomes obsolete after some time resulting in drop in correctness of the system. On the other hand, current online algorithms are not viable due to time constraints, or scalability issues with respect to large variable space or data size. There are two measures of concern here: accuracy and efficiency. Accuracy is the measure of correctness of system and efficiency is the measure of how the system operates. This includes response time, resources utilized, etc., whereas an off-line model affects the accuracy, online models affect the efficiency of the system.

We observed that the degradation of system is dependent more on the model’s ability to reflect the world adequately and less on the time passed since the inception of the model. The accuracy of the system remained healthy while the distribution of knowledge in input stream was similar to the knowledge of training data used to create the model. But degradation results in some cases quite rapidly when knowledge obtained from the input stream diverges from the model [Hassan et al. (2009b), Junejo and Karim (2007)]. This evolution of the world is the major cause of degradation of service.

We have seen self-optimization [Hassan et al. (2010)] or self-healing as a possible solution. However, self-optimization, or self-healing, properties by their very definition are not able to respond effectively to the evolution of the world [Berns and Ghosh (2009), Kephart and Chess (2003)]. Our argument is that, since the downgrade of correctness is due to minor changes in some parts of the world, the solution to these shortcomings should also focus on accommodating these changes by minimally tuning the model locally i.e. self-calibration. This will make these adaptations efficient enough to update the model at runtime resulting in a more accurate yet scalable system.
5.1.3 Personalization

Social media networks like Bibsonomy\textsuperscript{1}, delicious\textsuperscript{2}, Flickr\textsuperscript{3}, and YouTube\textsuperscript{4} create a folksonomy where people share their resources and assign tags to them in a collaborative manner. An important component of these sites is a tag recommendation system that helps the users in assigning labels to the resources i.e. articles, images, videos etc. that they share. This tag recommendation system also helps the site in better organization of resources because a recommended tag has less chances of being misspelled or being a new unique word every time.

Tag recommendation can be either personalized or non-personalized. Personalized tag recommenders take into account a user’s personal tagging behavior, also called personomy, while recommending her tags for a post. Whereas the non-personalized tag recommenders do not use the personomy and recommend tags that are popular in the history. Personalization looks reasonable as users’ choices vary from each other. But these choices may vary over time and season for the same user showing the need for popular tags. We study the role of personalization in detail in this work.

There are two main techniques of recommending tags. Tag recommendation using resource contents, and tag recommendation using collaborative filtering (CF). Content based Recommendation takes the knowledge from the attributes and properties of the user and items such as title or description of the post or image. Whereas in Collaborative Filtering a user is suggested what other users with similar interest are tagging.

A. Hotho et. al. [Hotho \textit{et al.} (2006)] propose a graph based tag recommendation algorithm called adapted PageRank. Users, resources and tags make a tripartite graph whose link structure is used for recommendations.

A simpler approach can be recommending the most popular tags in the entire dataset. Popularity is being judged by frequency of occurrence. The weakness here is that we have a fixed set of tags for all users (that is why it is non-personalized) which does not seem a very attractive choice.

\begin{itemize}
  \item \textsuperscript{1}http://www.bibsonomy.org
  \item \textsuperscript{2}http://www.delicious.com
  \item \textsuperscript{3}http://www.flickr.com
  \item \textsuperscript{4}http://www.youtube.com
\end{itemize}
in general. In literature, we find personalized tag recommendation outperforming non-personalized recommendation. Rendle et. al. present a personalized graph based tag recommendation algorithm called Pairwise Interaction Tensor Factorization (PITF) [Rendle and Schmidt-Thieme (2010)]. In PITF, a user’s personomy i.e. tagging history, gets more weight as compared to other tags.

The extent of personalization in FolkRank [Hotho et al. (2006)] is less than PITF, but more than adapted PageRank. Personalized algorithms do not perform well when most of the users in folksonomy have small personomies.

This work analyzes the personomies in folksonomies. We study the connection of tagging history with the personalized or non-personalized recommendation. We do data analysis and study users’ tagging behavior. We come to the point that a folksonomy user’s personal history usage behavior keeps on changing across time and across platforms. Hence a personalized recommender performing good on one dataset may not perform good on other. On the basis of our observations regarding personal tag history usage, we propose a generic life cycle of a folksonomy user across all datasets.

5.2 Self-Optimizing the Tag Recommender

Our discriminative clustering based tag recommendation system will have to be updated from time to time to maintain its recommendation accuracy. This is because of changes in posting and tagging behaviors and additions to the tag vocabulary. Two questions arises while designing a self-optimizing system: (1) when should the recommendation system be updated? (2) how should it be updated?

The answer to the first question is easy. The recommendation system should be updated when its recommendation accuracy drops by more than a specified amount. The specified drop in accuracy can also be related to the number of recommendations after which an update is required, as demonstrated later in our experiments. In our context, update means re-building the clustering model again. The recommendation system can also be updated after a specified time interval. The administrator needs to specify the thresholds for accuracy drop and/or time interval.

The answer to the second question of how should the recommendation system be updated is
more involved. First, we note that re-building the clustering model takes time. The computational complexity of the discriminative clustering method is $O(NKI)$, where $N$ is the number of posts, $K$ is the number of clusters, and $I$ is the number of iterations. The clustering method converges satisfactorily in fewer than 15 iterations (see [Hassan et al. (2009b)]) thus removing $I$ from being a variable. This leaves $N$ and $K$ as the two key variables defining the re-building time. Second, as demonstrated in our experiments later, the recommendation accuracy of our discriminative clustering based approach depends nonlinearly upon both $N$ and $K$.

Given the above observations, we define an optimization problem that is solved to determine the best values of $N$ and $K$ for re-building the cluster model. The optimization problem can be described qualitatively as

$$\text{Maximize: Accuracy}$$

subject to: $\text{Time} < t$

where $t$ is an administrator specified constraint on clustering time. This is in general a nonlinear optimization problem with both accuracy and time dependent on $N$ and $K$.

We quantify the optimization problem empirically by learning the relationship of accuracy and time with their dependent variables ($N$ and $K$). Once this is done, the optimization problem is solved to find the optimal values for $N$ and $K$ given the time constraint that maximizes the recommendation accuracy. The time constraint $t$ is another parameter that is specified by the administrator.

### 5.2.1 Experimental Setup

**Data and their Characteristics**

We evaluate our approach on data made available by the ECML PKDD Discovery Challenge 2009 [Eisterlehner et al. (2009)]. The data are obtained from dumps of public bookmark and publication posts on BibSonomy [BibSonomy (2009)]. The dumps are cleaned by removing spammers’ posts and posts from the user dblp (a mirror of the DBLP Computer Science Bibliography), and by various text normalizations.

The post-core at level 2 data are obtained from the cleaned dump (until 31 December 2008) and
contain all posts whose user, resource, and tags appear in at least one more post in the post-core data. The post-core at level 2 contain 64,120 posts (41,268 bookmarks and 22,852 publications), 1,185 distinct users, and 13,276 distinct tags.

For this work, we use the \textit{tas} table of the post-core at level 2. This table contains the tag assignment (who attached which tag to which resource). Key fields of \textit{tas} include: user ID, tag, content ID, and date. We process the posts in the order in which they are posted.

**Evaluation Criteria**

The performance of tag recommendation systems is typically evaluated using precision, recall, and F1 score (or \textit{FScore}), where the F1 score is a single value (harmonic mean) obtained by combining both precision and recall. We report the precision, recall, and F1 score averaged over all the posts in the testing set.

**5.2.2 Experimental Results**

In this section, we present an analysis of our self-optimizing discriminative clustering based tag recommendation system. We present results demonstrating the effectiveness of discriminative clustering based tag recommendation. We evaluate its characteristics and discuss the implementation and evaluation of our self-optimization strategy.

**Discriminative Clustering based Tag Recommendation**

The performance of the discriminative clustering method is evaluated on the post-core at level 2 data. We cluster the posts in the training set based on the tags assigned to them. After clustering and ranking of tags for each cluster, we recommend the top 5 tags from the ranked list of the assigned cluster for each post in the test set. We report the average precision, recall, and FScore (used interchangeably with accuracy) values over the test set for the recommendation system. Unless stated otherwise, the training set contains the first 30,000 posts and the test set contains the remaining 34,120 posts.
Relationship of \( N \) and \( K \) with Time and FScore

We study the recommendation performance of the discriminative clustering approach by varying the number of posts \( N \) in the training set and the number of clusters \( K \). We find an almost linear relationship between clustering time and \( N \), and clustering time and \( K \). However, the dependence of clustering time on both \( N \) and \( K \) together is nonlinear. This is seen from the 3D surface plot of clustering time with \( N \) and \( K \) (Figure 5.1). We also verify the nonlinearity of this relationship by fitting first, second, and third degree polynomials to the data, finding that first and second degree polynomials produce large sum of squared errors (SSE) as compared to that produced by the third degree polynomial.

Figure 5.1: Clustering time versus both \( N \) and \( K \)

Nonlinear relationships exist between \( N \) and FScore and between \( K \) and FScore. More importantly, these nonlinear relationships are not always monotonic. Similarly, the relationship of FScore with both \( N \) and \( K \) is nonlinear in nature as demonstrated by the 3D surface plot in Figure 5.2. For this relationship also, we verify its nonlinearity by fitting first, second, and third degree polynomials, finding that a third degree polynomial fit produces a lower SSE.
Performance Variation With Number of Test Posts

After the tag recommendation system is built from training data it is used to suggest tags for new posts. The performance of the system on new posts is bound to change with time as posting behaviors change. We investigate this aspect of the system by evaluating its performance on test posts ranging in quantity from 1,000 to 34,000 (Figure 5.3). The variation in performance is clear from this figure. More interestingly, it is seen that FScore values drop sharply after reaching a peak at about 6,000 test posts. Analyzing the data we find the reason for this drop in performance: there are a couple of new users actively posting using a single tag (in German language). This observation highlights the need to monitor performance and to update the system when performance drops below a specified threshold.

When and How to Update the Recommendation System

Updating our tag recommendation system involves re-building the clustering model by running the discriminative clustering method. This will allow the recommendation system to adapt to changes and drifts in the data, and ensure continued high performance. Two questions arise in this respect: (1) when should we update our recommendation system? (2) how should we update it, i.e., what should be the values of the parameters of clustering (number of training posts \(N\) and number of \(K\))?
clusters ($K$))?  

Figure 5.3 helps us in answering the *when* part of the question. It is seen that the performance varies with time as new posts are seen. Thus, when the performance monitor reports an average FScore drop greater than the specified threshold, re-clustering is required. More specifically, the drop in performance should be monitored in the past $b$ recommendations. For instance, if in the last 1,000 recommendations the FScore drops by more than 2%, then the recommendation system needs to be updated by re-building the clustering model. Figure 5.4 shows the performance of the recommendation system over the *next* 1,000 posts when the system is updated (using fixed non-optimal re-clustering) and when it is not. An update is only done when the performance drops by more than 2% in the last 1,000 posts. Figure 5.4 shows that the average FScore of the system that is updated is about 5% higher than that of the system that is not updated. Non-optimal re-clustering uses all available posts seen so far, and $K$ is set to 200.

The second question of *how* to update the recommendation system requires deciding the parameters of clustering. That is, we have to choose the values of $N$ and $K$. Figure 5.2 has demonstrated that the values of $N$ and $K$ can significantly impact the performance of the recommendation system. We need to maximize the performance by re-clustering, given limited time. The optimal values of $N$ and $K$ for re-clustering are found by solving a nonlinear optimization problem, as discussed in Section 5.2.2.
Figure 5.4: Comparison between original and re-clustering based recommendation systems

Nonlinear Optimization Model

The relationship of clustering time and FScore with both $N$ and $K$ is shown in Figures 5.1 and 5.2, respectively. These relationships are modeled by a third degree polynomial. After determining the third degree polynomials for FScore and time, we formulate the optimization problem as a nonlinear constrained integer programming problem. For presentation convenience in the following expressions $N$ is the number of training posts in thousands and FScore is given as a percentage.

Maximize $FScore = \begin{align*} 
&3.26 \times 10^{-5}N^3 + 1.35 \times 10^{-6}K^3 + 1.84 \times 10^{-7}NK^2 + 4.43 \times 10^{-6}N^2K - 0.0049384N^2 - 0.00079702K^2 - 0.00023509NK + 0.22396N + 0.15754K + 8.4041 \\
such that \\
&0.00036217N^3 + 9.20 \times 10^{-7}K^3 - 8.03 \times 10^{-6}NK^2 - 0.00011386N^2K - 0.042938N^2 - 0.00041314K^2 + 0.027492NK + 1.6426N + 0.047557K - 12.314 < t \\
&and \ 1 \leq N \leq N_m \\
&and \ 1 \leq K \leq K_m \\
&and \ N \ and \ K \ are \ integers.
\end{align*}$

81
In the above formulation, $t$ is the maximum available time allowed for re-clustering, $N_m$ is the maximum number of training posts in thousands, and $K_m$ is the maximum number of possible clusters (which is set to 300 in our experiments).

The nonlinear curve fitting and optimization problems are solved efficiently by using the MATLAB software. For nonlinear constrained optimization problem, Matlab’s function fmincon with default settings is used. fmincon uses trust-region-reflective algorithm for optimization by default. As an example, setting $t$ to be 200, $N_m$ to be 55 (i.e. 55,000 training posts), and $K_m$ to be 300, the optimal solution returned by MATLAB is $N = 55$ and $K = 175$.

![Figure 5.5: Performance of original, re-clustering, and self-optimizing tag recommendation systems](image)

**Performance of Self-Optimizing Tag Recommendation System**

We evaluate the performance of our self-optimizing tag recommendation system under several settings demonstrating its benefit. Figure 5.5 shows the average FScore of the system under five settings: (1) ORG: *Original* system without re-clustering, (2) FRC: *Fixed re-clustering* (whenever FScore drops by more than 2% in the last 1,000 posts) using all seen posts, $K = 200$, and $t$ is unbounded, (3) ORC1: *Optimum re-clustering 1* as in (2) but with optimal values for $N$ and $K$, and $t \leq 120$ seconds, (4) ORC2: *Optimum re-clustering 2* as in (3) but $t \leq 300$ seconds, and (5) ORC3: *Optimum re-clustering 3* as in (3) but $t$ is unbounded.

These results demonstrate the benefit of our self-optimization strategy. The self-optimizing
settings ORC2 and ORC3 significantly outperform the fixed re-clustering (FRC) and no update
(ORG) settings. Their FScore value of 23% is 6% higher than that obtained by the original setting
and 1% higher than that obtained by the fixed re-clustering setting. It is seen that for this relatively
small data the clustering time does not play a significant role in the results.

5.3 Self-Calibrating the Tag Recommender

5.3.1 Defining Self-calibration

Calibration is defined as:

**Definition 1.** A minimal tuning, filtration, or characterization of raw information about the sys-
tem’s environment to reduce the gap between system’s world view and the actual world.

We define a self-calibrating system as:

**Definition 2.** A system which is able to:

1. Identify the need for calibration by intelligently observing relevant raw information and the
   internal constructed system’s world view

2. Instrument appropriate calibration actions on the system to minimize the gap between system’s
   world view and the actual world.

Self-calibration is an ongoing process in which system first identifies the point where system
model and actual world diverges from each other. It then initiates processes to reduce this gap
whenever and however possible. This property, as is evident from definition and its application, is
not a goal in itself like self-optimization or self-healing. Rather it is an enabling property which
supports efficient self-healing and self-optimization by using observational traits of self-healing and
self-protection. This ultimately helps in planning and executing a boost in efficiency to provide
self-optimization property to the system.
5.3.2 When and What to Self-calibrate

There are two variants of calibrating system based on core demands of system. For a system concerned about quality of service (QoS) we can set a threshold of health $\tau$ such that when distance of a sub-component (estimated versus actual/required) breaches this threshold, re-calibration of this component is performed.

$$|\bar{j}_{M_i}^s - j_{M_i}^a| > \tau$$

Another possibility is that we can limit the resource utilization by putting a threshold at the number of calibrations that can be done in an iteration. In this method the components are sorted according to $\Delta_{\text{error}}$ and top $x$ are chosen for calibration where $x$ is dependent upon the available resources.

This type of calibration maintains the safety property that “bad things never happen to the system”. Moreover, through this system we can capture the system moving towards $D$ state and recover it before it goes to unhealthy state as described in [Ghosh et al. (2007)].

5.3.3 Self-Calibrating Tag Recommendation System

We proposed an automatic tag recommendation system for folksonomy based on discriminative clustering [Hassan et al. (2009b)]. A new document is first classified into a specific cluster and the top 5 tags of the cluster are recommended as possible tags for the document. The accuracy of the system however, was affected over time. As new ideas and concept emerge the tags and their relationships with documents also change.

To handle this accuracy drop due to distribution drift, we proposed to rebuild the prediction model by re-clustering [Hassan et al. (2010)]. To automate the task we proposed a self-managing mechanism for this process. A self-optimizing system though is able to keep a high level of accuracy, but such massive re-modeling step is an overkill. Especially when up to 65% of clusters remain intact and majority of the system is not affected by clustering. This points to the intuitive idea that over time not all information will change but rather some relations between document classification and tags would be re-ordered. What is required is not self-optimization but minimal self-healing of system.
We found that a large number of documents were being clustered correctly but the evolution of tags by users was not represented in the system. We implemented architecture 1 in this scenario. Figure 5.6 shows resulting f-score with and without self-calibration. Red line shows regular results when system allows cluster performance to degrade. Green line shows self-calibration results. It can be seen that between example 5 and 10 self-calibration identifies a need for calibration. After the calibration step accuracy of cluster is restored until 50th example when another round of calibration is done. The accuracy of cluster was 0.35 before example 5 (Y-axis value). Without self-calibration the average accuracy goes down to 0.0197. However, self-calibration restores the accuracy 0.329. In addition we observed that this drop in accuracy, or its healing, did not affect the global accuracy by much. Without healing global accuracy was 0.1592 and with self-calibration it was 0.16. Such a minor change in accuracy at global level was not observable but at a sub-component level it was observable and correctable. It is evident that self-calibration of system can make the system self-healing.

5.4 Personalization in Tag Recommendation Systems

We analyze the user tagging history behavior in detail [Uddin et al. (2011)]. We segment the life cycle on the basis of history usage and user personal information. Our hypothesis is that as the number of posts grows, more personal and less popular tags are used by a user. For evaluation and experimentation, three famous algorithms PITF, FolkRank and adapted PageRank are used on three datasets: Bibsonomy, Flickr and Delicious. Besides these graph based recommenders, to study the impact of personalization on content based tag recommendation, a clustering based tag recommendation system is also investigated using Bibsonomy data. Results give an insight about

![Figure 5.6: Comparison of cluster accuracy with and without self-calibration](image)
how a user in a folksonomy behave. Following sections provide a brief description and motivation of the four tag recommendation algorithms that we have selected.

5.4.1 Adapted PageRank

Undirected triadic hyperedges of a folksonomy graph instead of directed binary edges of the Web graph stop the direct application of PageRank [Brin and Page (1998)] algorithm to folksonomies. Hence Hotho et al. [Hotho et al. (2006)] present the adapted PageRank algorithm. The underlying principle is same i.e. a post assigned with important tags by important users becomes important as well. Adapted PageRank was adapted to a tripartite undirected hyper graph which unfortunately makes the in-link and out-link useless. A recent improvement is suggested by Ramezani [Ramezani (2011)]. Since Adapted PageRank is not utilizing any user personal information for ranking the tags, therefore it is a non-personalized tag recommender.

5.4.2 FolkRank

FolkRank [Hotho et al. (2006)] is an improvement over adapted PageRank. It calculates the final score by subtracting the weights of preferred tags from the generic but popular tags. In contrast to one global ranking independent of any preferences produced by adapted PageRank, FolkRank provides topic specific rankings. Thus FolkRank is personalized as compared to Adapted PageRank.

5.4.3 Pairwise Interaction Tensor Factorization (PITF)

Rendle et al. [Rendle and Schmidt-Thieme (2010)] present the factorization model PITF based on Tucker Decomposition model. It models the pairwise interactions between users, items and tags. Learning in PITF is done using Bayesian Personalized Ranking (BPR) criterion thus making it a personalized recommender system. PITF is a well known algorithm that won the ECML/PKDD discovery challenge 2009 for graph based tag recommendation.

The choice of the above three tag recommenders is done in order to test the behavior of increasing personalization on tag recommendation quality. Adapted PageRank is not personalized, FolkRank is more personalized than Adapted PageRank, and PITF is the most personalized of all three.
5.4.4 Tagging by Discriminative Clustering

In contrast to the three algorithms above, instead of using the link and node structure only, this algorithm [Hassan et al. (2009b)] utilizes the textual content of the shared resources to recommend tags. Posts are grouped based on the tags assigned to them and their content terms. The clustering method tries to maximize the sum of discrimination information provided by the posts, and outputs a list of weighted discriminating terms [Hassan and Karim (2012)]. This list and the user’s history of tags are used to generate the final recommendations.

In next section we describe the datasets and their characteristics in detail on which these recommenders are evaluated.

5.5 Datasets and their Analysis

We study three major dataset from folksonomies around the Web 2.0. The features are given in the table 5.1. For Flickr and Delicious datasets, we draw around 322000 number of tag assignments. This was strictly due to our system limitation and time constraint to perform experiments. Each tag assignment was drawn in order of natural occurrences. So data selection was not done on user specific selection criteria or some user specific demographics. The data sets are cleaned by applying same preprocessing done in ECML Challenge 2009. We observe that Delicious and Flickr datasets are closer to real world as they contain users having one or more posts. For ECML data (data of Bibsonomy), at least two posts for each user are present i.e. post core 2 data. Flickr is a “narrow folksonomy” [Vander Wal (2004)] therefore Flickr tags represent a very personal character of the user. Conversely, social bookmarking sites like bibsonomy and delicious have a reasonable number of users sharing same item or bookmark. General characteristics are shown in table 5.1. User behavior for each dataset has different background associated with them.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Users</th>
<th>Items</th>
<th>Posts</th>
<th>Tags</th>
<th>Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bibsonomy</td>
<td>1185</td>
<td>22389</td>
<td>63628</td>
<td>13276</td>
<td>248494</td>
</tr>
<tr>
<td>Flickr</td>
<td>6222</td>
<td>90834</td>
<td>90834</td>
<td>30996</td>
<td>321977</td>
</tr>
<tr>
<td>Delicious</td>
<td>3312</td>
<td>118009</td>
<td>153598</td>
<td>22134</td>
<td>321905</td>
</tr>
</tbody>
</table>
In a real dataset, firstly, we have a huge percentage of users which have less information in terms of their posts and tags (Figure 5.7). Often a content based solution is advised for such a user. On the contrary, there are high history users as well in real dataset but they are less in number. Secondly, we have items duplicated by some users assigning different tags to the same item. We see that items have quite different duplication factor in each of three datasets (table 5.2) while a same power law distribution is observed all the data sets. This implies that repeated items are more likely to be tagged differently by users. This will simply add more unique tags assigned to an item by different users. An item which has appeared only once may reflect a cold start for that particular item.

A personalized algorithm assumes that a user may use her personal history in future. PITF [Rendle and Schmidt-Thieme (2010)] is personalized to a user and gives more preference to personomy, a user’s tagging history, while ranking tags. A person may use tags from his history more as his number of posts increases. Over period of time, it can be observed that the people who have posts less than ten and have not used history yet, may start using history as they post more. We find that there are significant numbers of low posters in all three datasets. We assume personalized

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Bibsonomy</th>
<th>Flickr</th>
<th>Delicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item Duplication</td>
<td>68.72%</td>
<td>0%</td>
<td>13.27%</td>
</tr>
</tbody>
</table>
recommendation may not be favorable to such users and in more real scenario; prediction quality of overall system may suffer.

5.5.1 User History Strength and User Tag Gain

To test this hypothesis that personalization is not favorable to low posters, we have defined three different measures on user tag history usage: User History Strength (UHS), User Tag Gain (UTG) and User Tag Strength (UTS). User History Strength is the number of tags a user has used from his history. User Tag Gain is the number of tags a user has never used after tagging a post once. For instance, in table 5.3, user 869 has used tags repeatedly from history with UHS value 1099 and UTG value 51. User 869 can be treated as a person using more of his history and is less likely to add more new tags. But a user such as 10 with greater UTG value and smaller history usage is more inclined to use other sources of tags. User Tag Strength is the number of a user’s unique tags. Table 5.3 shows average tag assignment per post as ratio and Unique Tags as unique number of Tags referenced in UHS.

The history usage in Flickr as compared to Delicious and Bibsonomy is very low. This can be attributed to the Flickr background that every image is personal to user and cannot be shared like bookmarks or articles as in Delicious.

We observe that UTG and UHS are showing strong correlation in three datasets, table 5.4. This implies that it can be generalized that users tend to use history more as they contribute more in sharing. Also, as a user’s post count increases, his personal history will contain more tags which are assigned only once. Over all, a high history user tends to possess a large unique list of tags out of which top-n are to be recommended. Clearly, a tag recommendation for high history user may not

<table>
<thead>
<tr>
<th>User Id</th>
<th>UTG</th>
<th>UHS</th>
<th>Unique Tags</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>869</td>
<td>51</td>
<td>1099</td>
<td>41</td>
<td>0.32</td>
</tr>
<tr>
<td>872</td>
<td>1</td>
<td>123</td>
<td>12</td>
<td>0.33</td>
</tr>
<tr>
<td>17</td>
<td>287</td>
<td>742</td>
<td>139</td>
<td>3.4</td>
</tr>
<tr>
<td>18</td>
<td>21</td>
<td>6971</td>
<td>86</td>
<td>0.08</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.3: users’ tagging behavior in Flickr
Table 5.4: Correlation between UTG and UHS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Bibsonomy</th>
<th>Flickr</th>
<th>Delicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation</td>
<td>0.8443</td>
<td>0.4189</td>
<td>0.5933</td>
</tr>
</tbody>
</table>

give better performance, if no other information, like temporal analysis, is not incorporated [Lathia et al. (2010)]. In our study, we have not touched temporal aspects of user tag history usage and it has been left for future work. For analysis on high history users, we collected the stats of users who have consulted history more than ten times. Intuitively, it is more likely that high posters are high history users, table 5.5.

Comparing to Bibsonomy and Delicious data sets, Flickr has less high posters. Overall, there are at least 11.31% users who have more than ten unique tags in their history. We can conclude from above discussion that user tag history usage behavior persists in all datasets high posters.

### 5.5.2 Life Cycle of a Folk

On the basis of history usage, we have generalized folksonomy user life cycle, figure 5.8. A folkson-

![Figure 5.8: Life cycle of a Folksonomy User](image)

Table 5.5: Importance of UTS in High History Users

<table>
<thead>
<tr>
<th>Dataset</th>
<th>%UHS&gt;10</th>
<th>%UHS&gt;10</th>
<th>%UHS&gt;10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UTG&gt;0</td>
<td>UTS&gt;0</td>
<td></td>
</tr>
<tr>
<td>Bibsonomy</td>
<td>53.84</td>
<td>51.73</td>
<td>31.81</td>
</tr>
<tr>
<td>Flickr</td>
<td>35.47</td>
<td>25.70</td>
<td>11.31</td>
</tr>
<tr>
<td>Delicious</td>
<td>57.52</td>
<td>54.26</td>
<td>30.25</td>
</tr>
</tbody>
</table>
omy user becomes more interested in sharing as number of posts grows. We partition users into Beginners, Followers and Leaders on the basis of tagging history and number of posts. Users with less than 10 posts have less UHS, therefore labeled as Beginners. Users with more than 200 posts have High UHS, they are labeled as Leaders. Users with less than 200 posts and more than 10 posts have medium history usage hence labeled as Followers. These threshold values are taken by intuition and experiment, and the effect of change in these values is quite predictable. Increasing these values will in essence support the personalized recommenders. The terms Beginners, Followers and Leaders are considered with the concept of collaborative filtering (CF) in mind, as it assumes to follow the word of the mouth. The life cycle begins with start of a new user in a folksonomy where in CF we have cold start problem as system has no prior information about the user. The user keeps on posting/sharing until she becomes a Follower and may further move to become a Leader. Our analysis implies that Beginners have lowest average history usage as compared to Followers and Leaders. Followers have lower history usage than Leaders. In a real time scenario, a Leader may become a Follower, if he stops posting items. Similarly, a Follower can become a Leader by posting more frequently and by using more tag history.

During these stages, different but a generalized pattern of tag recommendations are observed. When the user is in the Beginner stage, system has little or no information about the users. The recommender cannot suggest from the personal history only, therefore popular tags are suggested more or any content based recommendations are used. When user is a Follower, she may have a potential to become a Leader, and if she does, a recommender needs to use her tag history or other tags from similar Leaders. A Follower needs recommendation with a mix to the personal and popular tags. A Leader uses her history most with a small proportion to the popular tags. Overall, our recommendation slides between personal and popular recommendation. Figure 5.9 shows that each stage user has different general behavior of choosing tag from their history. Life cycle division into stages will help in focusing a more adaptive mix of recommendation rather than just for a particular niche. Collaborative Filtering based Recommendation algorithms has shown good performance for post core 5 or more datasets but they give poor predictions when no or little information is given [Symeonidis et al. (2010)]. Content information can be used to improve tag
suggestion for Beginners specially [Lipczak et al. (2009)].

![Figure 5.9: Unique Tag Gain in Life Cycle of a Folksonomy](image)

5.6 Experiments

We now present the details of experiments conducted on the three datasets: ECML/Bibsonomy, Flickr and Delicious using three recommenders: Adapted PageRank, FolkRank and PITF. Two versions of content based Tagging by Clustering method are devised i.e. Non-personalized and Personalized. This content based recommender is tested on Bibsonomy dataset.

5.6.1 Evaluation Criteria

We report F1 score, also known as F-measure, in our experiments. It is the harmonic mean of precision and recall and is a popular measure for evaluating tag recommendation algorithms. It ranges from 0 (worst) to 1 (best). The formula is:

\[ F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \]

where

\[ \text{precision} = \frac{|\{\text{tags}_{\text{actual}}\} \cap \{\text{tags}_{\text{predicted}}\}|}{|\{\text{tags}_{\text{predicted}}\}|} \]
and
\[
\text{recall} = \frac{|\{\text{tags}_{\text{actual}}\} \cap \{\text{tags}_{\text{predicted}}\}|}{|\{\text{tags}_{\text{actual}}\}|}
\]

5.6.2 Experimental Setup

We performed two experiments to understand the behavior of folks categorized as Beginners, Followers and Leaders. First experiment is done on ECML/Bibsonomy dataset making sure that the train set remain postcore 2. Train and test split for Beginners are drawn by randomly drawing 1185 posts from ECML/Bibsonomy dataset and making sure at least one post remains in training set. For Leaders, top ten Leader posts are drawn and used to draw 1185 posts. For Followers, posters greater than five and less than Leader threshold are kept.

Second experiment was done on all three datasets by making Beginners, Followers and Leaders’ specific splits. We find out threshold on the basis of history usage in all these datasets. Beginners with less than 10 posts, Leaders with more than 200 posts and Followers with more than 10 and less than 200 posts are used as thresholds. The effect of these thresholds is discussed in results section (section 5.6.3).

We use parameter settings as discussed in the paper [Hotho et al. (2006)]. We tuned values of weight in adapted PageRank (0.0,0.9,0.1) giving best possible results empirically for each dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Dataset</th>
<th>Bibsonomy</th>
<th>Flickr</th>
<th>Delicious</th>
</tr>
</thead>
<tbody>
<tr>
<td>PITF</td>
<td>Beginner</td>
<td>0.2217</td>
<td>0.2924</td>
<td>0.1735</td>
</tr>
<tr>
<td></td>
<td>Follower</td>
<td>0.2568</td>
<td>0.3911</td>
<td>0.2477</td>
</tr>
<tr>
<td></td>
<td>Leader</td>
<td><strong>0.3589</strong></td>
<td><strong>0.4021</strong></td>
<td>0.2207</td>
</tr>
<tr>
<td>FolkRank</td>
<td>Beginner</td>
<td>0.2475</td>
<td>0.3277</td>
<td>0.2026</td>
</tr>
<tr>
<td></td>
<td>Follower</td>
<td><strong>0.3363</strong></td>
<td>0.3363</td>
<td><strong>0.2921</strong></td>
</tr>
<tr>
<td></td>
<td>Leader</td>
<td>0.3389</td>
<td>0.3891</td>
<td>0.2511</td>
</tr>
<tr>
<td>Adapted PageRank</td>
<td>Beginner</td>
<td>0.2524</td>
<td>0.3263</td>
<td><strong>0.2084</strong></td>
</tr>
<tr>
<td></td>
<td>Follower</td>
<td>0.2607</td>
<td>0.2607</td>
<td>0.2217</td>
</tr>
<tr>
<td></td>
<td>Leader</td>
<td>0.1822</td>
<td>0.3802</td>
<td>0.2237</td>
</tr>
</tbody>
</table>

Table 5.6: F1 Score for top 5 tag recommendations by graph based algorithms on three subdivided datasets
5.6.3 Results and Discussion

The objective of first experiment was to tune the threshold for the Beginners, Followers and Leaders. It was made sure that Beginners will be a major portion of the test set. We also verified the optimal parameter tuning of PITF for ECML/Bibsonomy dataset. We find that settings discussed in paper are best for all folk types i.e. Beginners, Followers, and Leaders. This experiment helped us to achieve better threshold for folk type splits. We performed at least five experiments and reported average in the figure 5.10. We can clearly observe that personalization varies significantly in each folk type. We can relate it to the data analysis done in section 5.5. Since less number of posts and less average history usage of Beginners are observed, the performance of personal history based algorithm is lowest. This strong correlation exists as we observe PITF gives systematically better performance from Beginners to Leaders.

The results clearly show the personalization works more for Leaders and lowest for Beginners, table 5.6.

In another experiment using PITF, we evaluated that high unique tag strength determines the $k$ reduction factor for the dataset. Reducing dimensions to 64 or 256 for ECML (Bibsonomy) dataset a little improvement in precision is shown [Rendle and Schmidt-Thieme (2010)]. Whereas performing similar experiment on flickr data set with same parameters, we have significant difference in quality between 64 and 256 flickr dataset. This seems reasonable as for more unique tags in personal...
history, a higher k factors is to be chosen to avoid loss of signals. Intuitively, for Beginners having more 50 unique tags will have poorest personal prediction performance. The Objective of second experiment was to analyze whether personal or popular suggestions works for each type of folks in all datasets. We have done five different random ‘leave one post out’ splits for each type of folk in all data sets.

We observe personal recommendations for Beginners are the poorest in quality where as the most popular recommendations by adapted PageRank give excellent results for Beginners. Similarly, for Followers FolkRank gives the best performance, although we observe personalization does improve here but not enough to overshadow the popular suggestions. We can imply a Follower folk overall uses more what others are using rather than what is more personal. We can also clearly say a Leader has not only done more posts, but also has been using more tags from others in his Follower stage. Hence, Personalization works for only Leaders seems justified.

From our data analysis, we know that all of these datasets have different backgrounds and different user intent. Flickr Dataset has no duplication factor. The personal choice of tags for all users seems to be like frequently used terms. A traveling image will be given general tags like ‘visit’, ‘travel’ and ‘party’. In Delicious and Bibsonomy as compared to Flickr, there are significant duplication of resources. A duplication means more possibility of assigning different tags. We observe PITF performs best for Follower and Leaders. We can relate it to table 5.5; for flickr only 11.31% of users are high posters and high history user, hence prediction quality just improves very little (0.3911 to 0.4021). For Beginners, we observe an insignificant difference in terms of quality in FolkRank and Adapted PageRank.

Table 5.7 shows the prediction results of content based tag recommendation for three subsets of Bibsonomy data [Hotho et al. (2009)] i.e. Beginners, Followers and Leaders. The results for ‘Non-

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Non-personalized</th>
<th>Personalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beginners</td>
<td>0.1612</td>
<td>0.1736</td>
</tr>
<tr>
<td>Followers</td>
<td>0.2521</td>
<td>0.2719</td>
</tr>
<tr>
<td>Leaders</td>
<td>0.2418</td>
<td>0.2564</td>
</tr>
</tbody>
</table>
The ‘Personalized’ column are calculated without looking the users’ tagging history. The ‘Personalized’ column shows the results when the content and clustering based tag recommender suggests tags from the user’s history too. About 12% of the tag assignment data is held for testing purpose for each of the three data subsets. For all the three subsets of Bibsonomy data, we find an improvement in prediction quality when the users’ personal tagging history is incorporated in recommendation process. Prediction performance for Followers as well as Leaders is significantly better than the prediction performance for Beginners.

The content based recommender shows a behavior different from all of the three graph based recommenders on Bibsonomy dataset, table 5.6. Its performance is not very good on beginners just like PITF but it reaches its best for Followers unlike PITF which shows best performance on Leaders. Adapted PageRank also performs best for Followers just like content based recommender but is not good for Leaders.

For this content based tag recommender, we can say that personalization is useful on Bibsonomy type of data where reuse of items is quite significant (68.72%, table 5.2).

We can ultimately conclude that tag and item duplication in dataset, history usage behavior such as average tag assignment of folk from history and global average tag assignment of folk play important role in determining what performance is achieved by a personal, popular or a mix of these two algorithms. We also believe that by allowing these features in a novel algorithm, one can achieve an adaptable and general tag recommendation system for any folksonomy. We also can see that folksonomies have common features, but user behaviors are attributed to these factors in total. For instance, we can judge that folks in Followers having more UHS and UTG have potential to become Leaders in future.

5.7 Conclusion

In this chapter, we propose and evaluate a self-optimizing discriminative clustering approach for tag recommendation in social bookmarking systems. The motivation behind making tag recommendation systems self-optimizing is to ensure continued high recommendation performance with minimal administrator input. We use an efficient discriminative clustering based approach for tag
recommendation. This recommendation system is then made self-optimizing by developing an optimization formulation that seeks to maximize the recommendation accuracy by finding the optimal values for number of training posts and number of clusters to use. To develop this formulation, we study and analyze the performance of the recommendation system on real-world data. In particular, we study the effect of changing clustering parameters on clustering time and recommendation accuracy. The relationships between clustering time and recommendation accuracy with the clustering parameters are determined empirically. We find that our self-optimization strategy can increase recommendation accuracy significantly with minimal administrator input.

For content driven systems where the new information from the outside world is continuously received the constructed model of the world gets obsolete with time. Taking the system to an offline mode to reconstruct the model is not a viable solution for time-critical systems. Reconstruction of the model using self-optimization or self-awareness is also inefficient as the process of reconstruction of the model is expensive and time consuming. However, through self-calibration, the model is updated with minimal changes.

In systems that receive a lot of content e.g. spam filtering, tag recommendation etc., much of the information is useless or false positive. Therefore, self-calibration also aims to mitigate the inaccurate information and only accept statistically significant information for the model tuning. We believe that for systems that use AI techniques and are heavily content-driven, self-calibration will provide an important missing property in making these systems self-managing.

Personalized or non-personalized techniques in recommendation systems may both suffer due to different user behaviors in folksonomies. We discover a general life cycle of a user in a folksonomy comprising of Beginners, Followers and Leaders on the basis of item/tag count and history usage behavior. We find measures UHS, UTG and UTS based on a user’s tagging history. We observe that these measures and the duplication factor of tags and items can play significant role in performance of prediction quality of recommender system.

We experiment on three social network datasets i.e. Bibsonomy, Flickr and Delicious using three graph based tag recommendation algorithms of varying personalization degree: Adapted PageRank, FolkRank and PITF. Personalized recommendation (PITF) gives good quality recommendations
for Leaders, but for Beginners non-personalized recommendation (Adapted PageRank) works best. Our experiments verify that a personalized algorithm may not perform good for all datasets in general.

Results for content based algorithm that uses clustering for tag recommendations are calculated on Bibsonomy dataset, showing the positive role of personalization.
Chapter 6

Web Surfing Behavior Prediction and Clustering

As stated in the beginning of the thesis, we present novel text clustering methods and their application to the content and usage types of Web mining in this thesis. The major focus of previous chapters (3 to 5) was on Web content mining. Although Web usage has been seen implicitly while discussing the tag recommendation systems, this chapter concentrates on Web usage mining explicitly. The popular problem of Web surfing behavior or path prediction is studied with our proposed solutions. Impact of clustering the surfing history using our Seq-Clust algorithm prior to prediction is analyzed.

6.1 Introduction

It is often stated that history repeats itself. Hence, so should Web surfers’ navigation paths as humans have the habit of repeating their actions. Similarly, it has been observed that people tend to form groups naturally based on their interests and behaviors. These observations suggest that behavior clustering may improve Web surfer behavior prediction. Web surfer behavior prediction is the process of determining key navigation patterns of users in the future given their navigation path history. Such predictions have immense commercial value as the Web evolves into a primary medium for marketing and sales for many businesses. Web-based businesses seek useful users’
patterns to help identify promising events, potential risks, and make strategic decisions.

Web surfer behavior modeling and prediction has been a popular research topic. Over the years numerous approaches have been proposed for solving various aspects of the problem with varying degrees of success. This is a complex machine learning problem that requires careful consideration from the technical and practical point of views. We consider the problem of predicting page categories visited by users in the first three positions of their navigation paths, where a specific page category refers to all Web pages on a particular theme or topic (e.g. sports, news, education). This is a simpler problem that can be modeled and interpreted more readily as compared to the general Web surfer behavior prediction problem [Nguyen (2007)]. Predicting multiple ordered values, which is typically known as sequence prediction, is of practical interest in many areas besides Web mining, e.g., forecasting the weather of next three days, predicting average stock value in next three weeks, etc.

Behavior clustering is commonly employed for user segmentation and navigation pattern visualization [Kim (2007), Cadez et al. (2003)]. Its use in behavior prediction has not been investigated extensively. In particular, the following questions need to be addressed: Does behavior clustering improve behavior predictions? Are clustering-based predictions more efficient? What is the efficiency-effectiveness trade-off of predictions done with and without clustering? In this chapter, we investigate these questions by evaluating the performance of generative and discriminative prediction models when they are built from the entire history of navigation paths and when they are based on behavior clusters of navigation paths. Our generative models are based on Markov chain and Bayesian classification, whereas our discriminative models are built using Support Vector Machine (SVM). We perform behavior clustering by adapting the $k$-modes clustering algorithm [Huang (1998)] with a new similarity measure and a new strategy for selecting initial clusters’ representatives. The performance of our models is evaluated on two real-world and publicly available data sets. The results show high prediction accuracy comparable to those reported in the literature. Furthermore, it is found that behavior clustering improves Web surfer behavior prediction in a few scenarios only but clustering-based predictions are often more practical than predictions based on unclustered data because of their efficiency. We also investigate several variations of our
clustering-prediction framework including varying the number of clusters, predicting based on top 
$r$ relevant clusters only, and predicting using cluster representatives only.

Key contributions in this chapter include (i) a simple clustering method for order-weighted nominal sequences, (ii) generative and discriminative models for navigation path prediction, (iii) impact of behavior clustering on Web surfer behavior prediction, and (iv) evaluation on two real-world data sets.

The rest of the chapter is organized as follows. Section 6.2 defines the Web surfer behavior prediction problem and introduces the mathematical notations. Our behavior clustering and prediction framework for predicting key Web navigation pattern is described in Section 6.3. Experimental evaluation of the models is given in Section 5. We conclude in Section 6.

### 6.2 Web Surfer Behavior Prediction Problem

Learning to predict the navigation paths of users given their navigation path history is commonly referred to as Web surfer behavior prediction. A navigation path is a sequence of Web page categories (or Web pages) that a user visits on a particular visit session. For a particular page category, a user may browse through a number of pages belonging to that category before terminating the session or moving on to another page category.

Let $\mathbf{D} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$ be the historical data of navigation paths for all users, where $\mathbf{p}_i$ identifies the $i$th navigation path and $N$ is the total number of navigation paths in the data. A navigation path $\mathbf{p}$ is defined by the vector

$$\mathbf{p} = [u, t, c_1, c_2, \ldots, c_n]$$

where $u$ is the user ID, $t$ is the start time of the session, and $c_i$ is the page category visited at position $i$ of the sequence. A navigation path can have a finite number $n \geq 1$ of page categories $c_i$ such that $c_i \neq c_{i+1}, \forall i \leq n - 1$ (i.e. a page category transition must occur at each position $i$ for $i < n$). In other words, a navigation path $\mathbf{p}$ is an $(\mathbf{x}, \mathbf{y})$ pair in which $\mathbf{x} = [u, t]$ defines the particulars of the navigation path and $\mathbf{y} = [c_1, c_2, \ldots, c_n]$ defines the sequence of page categories.
visited in the navigation path.

Instead of learning to predict the entire sequence of $c_i$ ($i = 1, n$), we focus on a simpler navigation pattern of practical value: predicting the first three page categories visited in a visit session. However, the proposed models can be extended easily to the prediction of any number of page categories. The choice of the first three page categories is an experimental convenience, especially when the focus of this work is on the impact of behavior clustering on surfing prediction rather than surfing prediction alone. In addition, same or similar problem setting of predicting first three or next three items only is also discussed in literature [Sauberlich et al. (2003), Nguyen (2007), Dembczynski et al. (2008), McKerrow (2008), Hassan et al. (2009a)]

Thus, given the above problem setting, the task is to learn to predict for known users the first three page categories visited, i.e., $c_1, c_2, c_3$. This is a machine learning problem in which the training data consist of navigation paths described by both $x$ and $y$ and the test data contain particulars of the navigation paths only, i.e., $x$ only and no $y$. Notice that the desired outcome is in general not a single label but a sequence of labels making this to be a sequence prediction problem.

6.3 Clustering-Prediction Framework for Web Surfer Behavior Prediction

We present a clustering-prediction framework for the solution of the Web surfer behavior prediction problem. This framework allows the use of both clustering and prediction methods for building Web surfer behavior prediction models. In this work, we formulate and evaluate models based on behavior clustering, models based on Bayesian or SVM prediction, and models based on a combination of both behavior clustering and Bayesian/SVM prediction. The latter combined approach, which captures the key ideas from the previous two approaches, consists of two steps. In the first step, unsupervised behavior clustering is done. The training data is grouped into clusters relying on the page category sequence information only (the $c_i, i = 1, n$ in $p$). In the second step, supervised learning is done from the behavior clustered information. We adapt the $k$-modes clustering algorithm for behavior clustering and use Markov chain based generative models and SVM based
discriminative models for prediction. These steps are described in detail in the subsequent sections. Figure 6.1 presents an overview of the proposed framework.

Figure 6.1: Overview of Clustering-Prediction Framework

6.3.1 Behavior Clustering

In the first step of our clustering-prediction framework for behavior prediction, the historical data of navigation paths, $D$, is partitioned into $k$ clusters based on the similarity between sequences of page categories (the $c_i, i = 1, n$ in $p$). Clustering is done by an adapted version of the $k$-modes clustering algorithm [Huang (1998)]. $k$-modes works similar to the iterative partitioning algorithm $k$-means, but in $k$-modes the centroid of a partition is the mode instead of the arithmetic mean, and a suitable similarity measure is used to better handle the categorical data. Our adaptations include strategies for selecting the parameter $k$ and the initial cluster representatives, and incorporation of a new similarity measure called $MatchScore$. 
Selecting $k$ and the $k$ Initial Paths

In most partitioning-based clustering algorithms, including the $k$-modes algorithm, two important decisions have to be made before their application: the selection of the desired number of clusters $k$ and the selection of the initial cluster representatives or seeds. Several strategies have been proposed for making these decisions. Pena et al. [Pena et al. (1999)] provide a comparison of four initialization methods for the $k$-means algorithm: random, Forgy, MacQueen, and Kaufman. Pantel et al. [Pantel and Lin (2002)] give an interesting cluster initialization method in their approach named CBC (Clustering By Committee). They first find well scattered tight clusters called committees and use them to construct the representatives of the initial clusters. Bradley et al. [Bradley and Fayyad (1998)] estimate the distribution of modes to provide better initial representatives for $k$-means clustering. Pelleg et al. [Pelleg and Moore (2000)] try to estimate the number of clusters by using a statistically based criteria that maximizes the model’s posterior probability.

Our strategy is simpler and relevant to the problem of behavior clustering of navigation paths. We set the maximum possible value of $k$ (denoted by $m$) to be the maximum number of navigation paths with zero pairwise similarity. These paths represent the most dissimilar paths in the navigation path history and hence are ideal for the initial seeds for clustering. This strategy defines both the number of desired clusters and the initial seeds for clustering. In our implementation, we experiment with $k = m$, $k \approx 0.5m$, $k \approx 0.25m$, and $k = 1$ (no clustering) by randomly selecting the $k$ most dissimilar navigation paths as initial cluster representatives.

**MatchScore Similarity Measure**

The choice of a similarity measure is critical to the effectiveness of a clustering algorithm. A similarity measure is considered good if it accurately captures the semantic of similarity or relatedness between the objects to be clustered. In this work, we propose a new similarity measure called MatchScore that is designed specifically for clustering of navigation paths. This similarity measure is motivated from the evaluation criterion used by the 2007 ECML/PKDD Discovery Challenge [Nguyen (2007)] and is similar to the order-weighted measure used by [Kim (2007)]. Given two
navigation paths $p_1$ and $p_2$, their $\text{MatchScore}(p_1, p_2)$ is the sum of scores for matching categories in the two paths, where the scores are based on the positions at which the categories match. Mathematically, MatchScore is defined as

$$
\text{MatchScore}(p_1, p_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \text{IsAMatch}(c_i, c_j) \times \min(s_i, s_j)
$$

(6.1)

where $n_1$ and $n_2$ are the lengths of the navigation paths $p_1$ and $p_2$, respectively, $c_i$ denotes the category visited at position $i$, and $s_i$ is the score for a match at position $i$. The function $\text{IsAMatch}(c_i, c_j)$ returns 1 when $c_i = c_j$ and this category has not been matched earlier; otherwise, the function returns zero. This means that once a category pair has been matched (at same or different position) it is not included in the score again.

The score $s_i$ gives greater weight to a match in the beginning of the path. In our work, we consider matches in the first three positions, and define $s_i$ as 5, 4, or 3 for $i$ equal to 1, 2, and 3, respectively. With this definition, MatchScore can range from 0 (no similarity) to 12 (maximum similarity). An example of MatchScore calculation is given in Table 6.1.

<table>
<thead>
<tr>
<th>Categories in $p_1$:</th>
<th>9</th>
<th>17</th>
<th>15</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categories in $p_2$:</td>
<td>9</td>
<td>15</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

$\text{MatchScore}(p_1, p_2) = 1 \times \min(5, 5) + 0 \times \min(5, 4) + 0 \times \min(5, 3) + \allowbreak 0 (\text{already matched}) + 0 \times \min(4, 4) + 0 \times \min(4, 3) + \allowbreak 0 (\text{already matched}) + 1 \times \min(3, 4) + 0 \times \min(3, 3)$

$= 5 + 3 = 8$.

This similarity measure is able to capture the fact that category matches in the beginning of visit sessions are more important and valuable than those occurring later in the session. Furthermore, it provides intuitive solutions to category matches that do not occur at the same position and for repeating categories in the navigation paths. Note that other existing measures like Kendall’s rank correlation and Spearman’s rank correlation [Kendall and Gibbons (1990)] are not suitable here as the problem is different from simple ranking. These ranking coefficients are unable to
capture the order-weighted semantics of navigation paths. Kendall tau distance checks the pairwise disagreements between two lists but does not give scores to partial matches as precisely as our MatchScore. Simple matching does not help either as the matches in early positions are considered more important here and should be given higher weight.

**Seq-Clust:** Adapted $k$-Modes Clustering Algorithm

We use a modified version of $k$-modes clustering algorithm to group together the navigation paths exhibiting similar surfing behavior. We cluster with different values of $k$ ($1 \leq k \leq m$), where $m$ is the maximum number of paths in the training data with zero pairwise MatchScore. These $m$ navigation paths are taken as initial cluster representatives or modes. The $k$-modes algorithm is executed starting from these initial cluster representatives adopting the MatchScore similarity measure. The clustering concludes the first step of our clustering-prediction framework. It produces $k$ clusters each represented by its representative navigation path, representing most frequent behavior of the complete cluster. Algorithm 2 states the steps of our Seq-Clust algorithm.

The clustering procedure is stopped when the overall MatchScore, or $ClusteringScore$, remains unchanged from one iteration to the next. The overall MatchScore is defined as follows:

$$
ClusteringScore = \sum_{i=1}^{k} \sum_{p \in L_i} MatchScore(p, d_i)
$$

where $p$ is a navigation path in cluster $L_i$ and $d_i$ is the mode of cluster $L_i$ (i.e. the most frequent sequence in cluster $L_i$).

Figures 6.2 and 6.3 verify the correctness of the clustering algorithm. They show the average

| Algorithm 2 Seq-Clust: Sequence Clustering using Adapted $k$-Modes |
|-------------------|-------------------|
| **Require:** $D$ (database containing $N$ navigation paths), $k$ (no. of clusters) |
| 1: choose $k$ unique navigation paths as the initial cluster seeds |
| 2: **repeat** |
| 3: (re)assign each navigation path to the cluster to which it is the most similar, based on the mode of the navigation paths in the cluster |
| 4: update the cluster modes, i.e., calculate the mode of the navigation paths for each cluster |
| 5: **until** overall MatchScore maximizes |
| 6: **return** A set of $k$ clusters that maximizes the overall MatchScore or $ClusteringScore$ |

106
Figure 6.2: Average intra-cluster similarity for different values of $k$ (ECML data)

Figure 6.3: Average intra-cluster similarity for different values of $k$ (KDD data)
intra-cluster similarity of each cluster for different values of $k$ (the number of clusters). The average intra-cluster similarity of a cluster is the average similarity of data points in the cluster from the cluster’s representative (mode). By definition of clustering, increasing the value of $k$ should result in more homogeneous clusters. This trend is highlighted in these figures. The average intra-cluster similarity increases with the increase in $k$ for two different Web navigation data sets. This verifies that our algorithm is successful in grouping similar navigation paths in the same cluster. The data sets utilized in these figures are described in Section 6.4.1.

After clustering the navigation path history, we proceed to the behavior prediction step.

### 6.3.2 Behavior Prediction

Predicting the sequence of page categories visited by users is valuable for strategic and operational decisions. Usually, only the first few positions in the sequence are useful as long sequences tend to drift. Two popular types of classification and prediction models have been used in practice: generative and discriminative. Generative models try to learn the posterior probability distribution by first estimating the probability distribution of the data. On the other hand, discriminative models try to learn a discriminant function directly from data and use this discrimination information to classify new data points. Each type of models has its pros and cons. We select the most representative models/methods from each type so as to ensure a fair evaluation of the impact of behavior clustering on prediction. Section 6.3.2 presents Bayesian and Markov models that are generative in nature, while Section 6.3.2 presents a SVM based model which is discriminative in nature.

**Bayesian Prediction**

In this section, we present a Bayesian model for predicting the first three page categories visited by a user in a visit session. The model is formulated for clustered navigation path data and its applicability to the entire (unclustered) data is highlighted in the presentation.

We assume that the page categories at positions 1, 2 and 3 form a Markov chain. The first category starts this chain and this category is determined by a Bayes classifier trained on the navigation path data. The second and third categories are decided by considering the Bayes classification at
that position and the transition probability from the previous position. The reason for selecting the first-order Markov model or Markov chain over higher order models is two-fold: (1) The problem involves the prediction of only the first three page categories for which a first-order Markov model is sufficiently accurate. (2) The first-order Markov model is computationally efficient as compared to higher-order Markov models.

According to the Bayes rule, the posterior probability of page category $C_j$ visited in position $j$ ($j = 1, 2, \text{or} 3$) of a visit session $x$ in cluster(s) $L$ is given by

$$P^B(C_j|x, L) = \frac{P(x|C_j, L)P(C_j|L)}{P(x|L)} \quad (6.3)$$

The conditioning on $L$ emphasizes that the probabilities are estimated from the data in $L$. This can be the data (navigation paths) in the most relevant cluster, in top $r$ most relevant clusters, or in the entire data (all clusters). With this notation, the same formulation can be used for prediction models built with and without clustering. The relevance of a cluster to an input $x$ is determined by the frequency with which the input occurs in the cluster.

The most probable page category visited at the start of the sequence ($C_1 = c_1$) is given by

$$c_1 = \arg\max_c P(x|C_1 = c, L)P(C_1 = c|L) \quad (6.4)$$

This fixes the start state of the Markov chain. The subsequent states can be found by combining the predictions of the Bayes classifier (Eq. 6.3) and the Markov model. According to the Markovian property, for a given visit session $x$ the posterior probability of page category $C_j$ visited in position $j (j = 2, 3)$ depends only on $C_{j-1}$ and can be expressed as

$$P^M(C_j|C_{j-1}, x, L) = \frac{P(C_j \land C_{j-1} \land x \land L)}{P(x \land C_{j-1} \land L)} \quad (6.5)$$

$$= \frac{P(x|C_j, C_{j-1}, L)P(C_j|C_{j-1}, L)}{P(x|C_{j-1}, L)} \quad (6.6)$$
Using Eqs. 6.3 and 6.6, the page category \( (C_j = c_j) \) visited at position \( j (j = 2, 3) \) is given by

\[
c_j = \arg \max_c P^B(C_j = c | \mathbf{x}, L) P^M(C_j = c | C_{j-1}, \mathbf{x}, L)
\] (6.7)

This equation is based on the assumption that the predictions of the Bayes and Markov models are independent. Note that the denominators in Eqs. 6.3 and 6.6 can safely be ignored in computation of Eq. 6.7.

The Bayesian models presented above are based on knowledge of the visit session, \( \mathbf{x} = [u, t] \), where \( u \) denotes the user ID and \( t \) identifies the time stamp of the session. When both \( u \) and \( t \) are considered in the models the naive Bayes assumption of conditional independence can be invoked to simplify the joint probability of \( u \) and \( t \) into the product of probabilities of \( u \) and \( t \).

The models (comprising of the various probabilities in the equations) are estimated from the historical training data by maximum likelihood estimation. Since all variables are observed in the training data, maximum likelihood estimates of the probabilities are equivalent to the respective frequencies in the data. Algorithms 3 and 4 provide the training and testing steps of our Bayesian prediction approach.

**SVM Prediction**

In this section, we present Support Vector Machine (SVM) approaches for Web surfer behavior prediction based on clustered and unclustered data. SVMs are popular discriminative classifiers that learn a maximum margin hyperplane for separating two classes in a feature space implicitly defined by a kernel function. SVMs have proven to be robust for many classification problems. Their applicability to sequence prediction problems requires adaptation of the original SVM for binary classification problems. We present a simple yet effective SVM-based approach for Web surfer behavior prediction that takes into account the dependence of page categories and navigation path.

---

**Algorithm 3** Behavior Prediction using Bayesian Models - Training Phase

**Require:** Clusters of training data, \((\mathbf{x}, \mathbf{y})\) pairs, where \( \mathbf{x} = [u, t] \), \( \mathbf{y} = [c_1, c_2, ..., c_n] \)

1: Estimate probabilities in equations 6.3 and 6.6  
2: **return** Estimated probabilities
Algorithm 4 Behavior Prediction using Bayesian Models - Testing Phase

Require: Estimated probabilities, Test data i.e. \( \mathbf{x} \) part only where \( \mathbf{x} = [u, t] \)

1: \textbf{repeat}
2: \hspace{1em} Take next test example \( x \)
3: \hspace{1em} Select \( L \), the most relevant cluster of \( x \)
4: \hspace{1em} Calculate \( c_1 \), as in equation 6.4
5: \hspace{1em} Calculate \( c_2 \) and \( c_3 \), using equation 6.7
6: \hspace{1em} \textbf{until} no test example left
7: \hspace{1em} \textbf{return} Predicted categories \( \mathbf{y} = [c_1, c_2, c_3] \)

As defined in Section 6.2, each navigation path is an \((\mathbf{x}, \mathbf{y})\) pair where \( \mathbf{x} = [u, t] \) contains the particulars of the navigation path and \( \mathbf{y} = [c_1, c_2, \ldots, c_n] \) is the sequence of page categories visited in the navigation path. To predict the first three page categories browsed in a navigation path, the following SVM-based approach is adopted:

1. A first SVM is trained to predict the first page category in the navigation path. The input is \( \mathbf{x} \) and the output is the page category visited at first position \( \mathbf{y} = [c_1] \).

2. A second SVM is trained to predict the page category visited at the second position of the navigation path. For this SVM, the input comprises of \( \mathbf{x} \) and page category \( c_1 \), while the output is the page category at second position \( \mathbf{y} = [c_2] \). When applying this SVM on test data, the page category at first position is determined by the first SVM described above.

3. A third SVM is trained to predict the page category visited at the third position of the navigation path. For this SVM, the input comprises of \( \mathbf{x} \), \( c_1 \), and \( c_2 \), while the output is the page category visited at the third position \( \mathbf{y} = [c_3] \). When applying this SVM on test data, predictions of the first and second SVMs are used as input to this SVM.

If navigation path data have been clustered, as described in Section 6.3.1, the above SVMs can be modified to include clustering information for cluster-based behavior prediction. This is done by adding a new input feature that identifies the most relevant cluster for the user. For example, in the second SVM described above, a third input feature is added whose value is set to the ID of the most relevant cluster for the user. Algorithms 5 and 6 provide the training and testing steps
of our SVM prediction approach.

6.4 Experimental Evaluation

We perform several experiments to analyze the impact of behavior clustering on Web page category prediction using Bayesian and SVM models. The evaluations include impact of number of clusters ($k$) formed by behavior clustering, number of most relevant clusters ($r$) used by prediction models, using time stamp in prediction models, and using cluster modes for prediction. We present results for two real data sets. In the subsequent sections, we describe the data and their important characteristics, the evaluation criteria, and the experimental results.

6.4.1 Data and their Characteristics

ECML Data

The 2007 ECML/PKDD Discovery Challenge [Nguyen (2007)] data set, henceforth referred to as the ECML data, was collected by Gemius SA 1, an Internet market research agency in Central and Eastern Europe, over a period of 4 weeks through use of scripts placed in code of the monitored Web pages. Web users were identified using cookies technology. The first 3 weeks of data are used for training while the last week’s data are reserved for testing. An example record of the data is given below.

<table>
<thead>
<tr>
<th>Path ID</th>
<th>User ID</th>
<th>Timestamp</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>1</td>
<td>1169814548</td>
<td>(7,3) , (17,3)</td>
</tr>
</tbody>
</table>

Path ID and user ID are the unique identifiers given to a visit session and a user, respectively. Timestamp is the Unix time at which the session started. Path is a sequence of (page category

---

1www.gemius.com, September 2013

Algorithm 5 Behavior Prediction using SVM Models - Training Phase

Require: Input data (in Weka format) including cluster attribute
1: Train $SVM_1$ on $x$ to predict $c_1$
2: Train $SVM_2$ on $x$ and $c_1$ to predict $c_2$
3: Train $SVM_3$ on $x$, $c_1$ and $c_2$ to predict $c_3$
4: return Prediction models: $SVM_1$, $SVM_2$ and $SVM_3$
Algorithm 6 Behavior Prediction using SVM Models - Testing Phase

Require: Prediction models: SVM$_1$, SVM$_2$ and SVM$_3$, Test data (in Weka format) i.e. $x$ part only
1: Predict $c_1$ using SVM$_1$
2: Predict $c_2$ using SVM$_2$
3: Predict $c_3$ using SVM$_3$
4: **return** Predicted categories $y = [c_1, c_2, c_3]$

ID, number of pages) pairs identifying the page category and the number of pages in that category visited.

Analyzing the data, we find that there are 20 unique categories. The three weeks training data comprise of 379,485 visit sessions whereas the one week test data contain 166,299 visit sessions. In addition, there are 4,882 distinct users having a non-uniform distribution in the data. The minimum and maximum number of visits by a user in the training data is 7 and 497, respectively, with an average of 77.7 visits per user. The minimum and maximum number of visits by a user in the test data is 1 and 215, respectively. Likewise, page categories are non-uniformly distributed in the data. Some of the categories are visited more frequently then the rest. ECML data set tracks users on several sites across a substantial geographical region and has a wider scope as compared to the second data set we use i.e. KDD data set.

For ECML data set, the maximum number of clusters is 19, i.e., $m = 19$ (the maximum number of dissimilar navigation paths). After partitioning the data into $k$ behavior clusters, we find that users spread over multiple clusters and this spread resembles a skewed normal distribution with long tail (Figure 6.4). It is seen from the figure that a vast majority of users are represented (present) in a few number of behavior clusters; however, few users are present in only one behavior cluster. Thus many users are exhibiting multiple behaviors.

**KDD Cup Data**

The KDD Cup 2000 [Kohavi *et al.* (2000)] data set, henceforth referred to as the KDD data, contain Web clickstream and purchase transactions collected by the Blue Martini Software application running at Gazelle.com, which used to sell legware and legcare products and is not operational now. The data spans over a period of three months – 30th January to 30th April, 2000. The first
two months’ data are for training purpose and the last month’s data are for testing and evaluation. This data contain more than 200 attributes including user’s demographic information, favorite options, session information, etc. Since we are interested in navigation path information only, we preprocess the data to the same format as the ECML data described above. For this purpose, Request Date, Request Date Time, Request Sequence, and Session Cookie ID attributes of original data are used. Every request template (the dynamic Web page requested by the user) in the original data is considered a separate page category (e.g. home, lifestyles, leg news, departments, vendor, etc). These categories are assigned unique numeric IDs. A category (request template) includes links to other categories and static Web pages which is used to form the sequence of page categories and views per page categories. The processed data contain 85 unique categories, 3,577 distinct users, 23,911 visit sessions for training, and 4,690 visit sessions for testing.

In this data also, users have a non-uniform distribution in the data. The minimum and maximum number of visits by a user in the training data is 3 and 744, respectively, with an average of 6.7 visits per user. The minimum and maximum number of visits by a user in the test data is 1 and 256, respectively. Likewise, page categories have a non-uniform data distribution with some categories being visited more frequently than others.

The maximum number of clusters for this data is 45 i.e. $m = 45$ (the maximum number of dissimilar navigation paths). After performing behavior clustering, the distribution of users across clusters is a bit different from the ECML data. Figure 6.5 shows that the majority of users are present in 2 or 3 clusters, i.e., the spread of users across clusters is somewhat narrower as compared to that for ECML data. This is because of the limited scope of this data with all pages belonging to one site and with most users starting surfing from one page.

### 6.4.2 Evaluation Criteria

We evaluate our models by computing a percent score as follows: For each predicted navigation path, the three predicted page categories are compared with the page categories in actual navigation path and a score is generated. This score is the sum of weights assigned to the three predicted
Figure 6.4: Distribution of users across behavior clusters for $k = 10$ and $k = 19$ (ECML data)

Figure 6.5: Distribution of users across behavior clusters for $k = 20$ and $k = 45$ (KDD data)
categories. The (average) percent score for predicting $N$ navigation paths is given by

$$\text{Percentscore} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{Score}(i)}{\text{MaxScore}} \times 100$$

where $\text{Score}(i)$ is the score for navigation path $i$ and $\text{MaxScore}$ is the maximum possible score for a prediction.

The calculation of the score is very similar to the calculation of our order weighted similarity measure discussed in Section 6.3.1. The difference here is that the number of categories in actual navigation path is not fixed to be three. For example, the score for a predicted path is calculated as follows. If the first, second, and third categories are predicted correctly, then assign weights 5, 4, and 3, respectively, to these positions. If a prediction is incorrect for the category at first position, then it is assigned a weight of 4 if that category occurs in the second position, 3 if it occurs in the third position, 2 if it occurs in the fourth position, 1 if it occurs in position five and beyond, and 0 if it does not occur. The weight assigned cannot be greater than the maximum possible for that position (e.g. the weight assigned to position 3 cannot be greater than 3). This evaluation criteria was also used by [Nguyen (2007)] for the evaluation of Web surfer behavior prediction.

For SVM models, in addition to computing the percent score as described above, the standard classification accuracy is reported for each page category in a navigation path.

### 6.4.3 Results and Discussion

#### Bayesian Prediction Results

Tables 6.2 and 6.3 show percent scores for predicting the first three page categories visited for ECML and KDD data, respectively. Prediction results are given for Bayesian models based on no clustering ($k = 1$) and three different clusterings, including when $k = m$. Moreover, prediction results are presented for models based on the top $r$ most relevant clusters (including $r = 1$ and $r = k$) for each user.

For ECML data (see Table 6.2), the best prediction is obtained when $k = 5$ and $r = 3$. However, this result is only marginally better than that produced without clustering ($k = 1$). In general,
increasing the number of relevant clusters used from \( r = 1 \) in cluster-based prediction improves performance with results for \( r \geq 4 \) being equal to or better than that obtained without clustering. This is consistent with the observation that the majority of users spread over four behavior clusters (see Figure 6.4). Thus, using four most relevant clusters in prediction yields the best results as higher numbers of relevant clusters often contribute to noise.

For KDD data (see Table 6.3), the best predictions are obtained when no clustering is done \((k = 1)\) or when a large fraction of relevant clusters \((r \geq 10)\) are used for prediction. Thus, a large fraction of data is required to achieve the best performance, and cluster-based prediction is less effective (as compared to that for ECML data). Moreover, as for ECML data, the best result is not significantly better than that obtained when a fewer number of relevant clusters are used in cluster-based prediction.

We experimented with maximum likelihood estimation of the models with and without add-one smoothing with no significant difference in performance.

**SVM Prediction Results**

Tables 6.4 and 6.5 show the SVM prediction results for ECML and KDD data, respectively. These tables give the percent accuracy for predicting the page category at each position (i.e. \( c_1, c_2, \) and \( c_3 \)) and the percent score for predicting the entire sequence. For cluster-based predictions, the number of clusters used are 19 and 45 for ECML and KDD data, respectively, and the most relevant cluster is included as an input feature. These results are obtained from implementing the first, second, and third SVM model (with and without clustering) using LibSVM [EL-Manzalawy and Honavar (2005)] with RBF kernel function.

As compared to the best results for Bayesian prediction, SVM prediction percent scores are slightly lower for ECML data and slightly higher for KDD data. For both data sets, however, SVM prediction is not impacted by incorporating the most relevant cluster input feature in the model.
Table 6.2: Bayesian prediction percent scores (ECML data)

<table>
<thead>
<tr>
<th>Relevant Clusters</th>
<th>$k = 19$</th>
<th>$k = 10$</th>
<th>$k = 5$</th>
<th>$k = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r = 1$</td>
<td>82.30</td>
<td>82.28</td>
<td>82.14</td>
<td>82.94</td>
</tr>
<tr>
<td>$r = 3$</td>
<td>82.93</td>
<td>82.96</td>
<td><strong>82.99</strong></td>
<td>-</td>
</tr>
<tr>
<td>$r = 4$</td>
<td>82.97</td>
<td>82.97</td>
<td>82.94</td>
<td>-</td>
</tr>
<tr>
<td>$r = 5$</td>
<td>82.96</td>
<td>82.96</td>
<td>82.94</td>
<td>-</td>
</tr>
<tr>
<td>$r = k$</td>
<td>82.94</td>
<td>82.94</td>
<td>82.94</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6.3: Bayesian prediction percent scores (KDD data)

<table>
<thead>
<tr>
<th>Relevant Clusters</th>
<th>$k = 45$</th>
<th>$k = 20$</th>
<th>$k = 10$</th>
<th>$k = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r = 1$</td>
<td>60.42</td>
<td>61.33</td>
<td>61.51</td>
<td><strong>62.08</strong></td>
</tr>
<tr>
<td>$r = 2$</td>
<td>61.11</td>
<td>61.96</td>
<td>61.35</td>
<td>-</td>
</tr>
<tr>
<td>$r = 3$</td>
<td>61.27</td>
<td>61.65</td>
<td>61.49</td>
<td>-</td>
</tr>
<tr>
<td>$r = 4$</td>
<td>61.78</td>
<td>61.82</td>
<td>61.93</td>
<td>-</td>
</tr>
<tr>
<td>$r = 5$</td>
<td>62.07</td>
<td><strong>62.08</strong></td>
<td><strong>62.08</strong></td>
<td>-</td>
</tr>
<tr>
<td>$r = k$</td>
<td><strong>62.08</strong></td>
<td><strong>62.08</strong></td>
<td><strong>62.08</strong></td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6.4: SVM prediction percent accuracies and percent scores (ECML data)

<table>
<thead>
<tr>
<th>Prediction</th>
<th>SVM</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Without Clustering</td>
</tr>
<tr>
<td>$c_1$</td>
<td>First</td>
<td>74.47</td>
</tr>
<tr>
<td>$c_2$</td>
<td>Second</td>
<td>46.70</td>
</tr>
<tr>
<td>$c_3$</td>
<td>Third</td>
<td>56.73</td>
</tr>
<tr>
<td>Average</td>
<td>-</td>
<td>59.30</td>
</tr>
<tr>
<td>Score (%)</td>
<td>-</td>
<td><strong>82.87</strong></td>
</tr>
</tbody>
</table>

Table 6.5: SVM prediction percent accuracies and percent scores (KDD data)

<table>
<thead>
<tr>
<th>Prediction</th>
<th>SVM</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Without Clustering</td>
</tr>
<tr>
<td>$c_1$</td>
<td>First</td>
<td>72.49</td>
</tr>
<tr>
<td>$c_2$</td>
<td>Second</td>
<td>20.84</td>
</tr>
<tr>
<td>$c_3$</td>
<td>Third</td>
<td>20.59</td>
</tr>
<tr>
<td>Average</td>
<td>-</td>
<td>37.97</td>
</tr>
<tr>
<td>Score (%)</td>
<td>-</td>
<td><strong>62.47</strong></td>
</tr>
</tbody>
</table>
Comparison and Discussion

Prediction results for ECML data have also been reported in [Dembczynski et al. (2008), Hassan et al. (2007)]. In this work, we train our models using the first three page categories only and our results are slightly lower than that reported in [Hassan et al. (2007)]. However, when we train the models on complete navigation paths, our results (with no clustering) are slightly better than those reported in [Hassan et al. (2007)]. Our results are slightly lower than those reported in [Dembczynski et al. (2008)] as well. However, our models are simpler and more efficient than the risk minimization and regression based approach developed by [Dembczynski et al. (2008)]. Moreover, none of the two approaches use or investigate behavior clustering for behavior prediction.

As seen from the previous section, behavior clustering is not improving behavior prediction significantly or consistently. To verify the statistical significance of behavior clustering on prediction, we perform the paired t-test at 0.05 level of significance. The normality of the data is verified using 1-Sample K-S test. The null hypothesis of the t-test is: prediction means of with and without clustering samples are equal. The results of the paired t-test reveal that at 0.05 level of significance we cannot reject the null hypothesis. Hence, clustering has no significant impact on behavior prediction, as determined for our selected real world data sets. Thus, behavior clustering, which is computationally expensive in itself, should not be performed if the only goal is to obtain better prediction models. However, it remains to be investigated whether this observation still holds when several months or years of historical data are available.

Recently, it has been reported that the practice of generating user navigation sessions using timeouts can break a single logical session into two [32]. This issue negatively affects the performance of any statistical model built from such data. More specifically, it muddles the user behavior at the start of the visit session. In additional experiments, we found out that this issue exists in our experimental data sets. For example, in the ECML data, sessions are terminated when a user remains inactive for 30 minutes. We found out that more than 15% of the sessions in this data are continuations of a previous session of the same user that started between 30 and 60 minutes ago. This discovery provides another reason for the insignificant impact of behavior clustering on the prediction of the first three page categories visited by users. The above observation suggests that
we should focus on position independent behavior prediction or highly probable sequence discovery. These are topics for future investigations and are beyond the scope of the current thesis.

**Incorporating Timestamp in Prediction Models**

All the results presented above are for the case when only the user ID, \( u \), is considered while building and applying the models. That is, the time stamp information, \( t \), of visit sessions has not been used. In this section, we investigate the impact of incorporating time stamp information (in addition to user ID) in Bayesian and SVM prediction when behavior clustering is not done (\( k = 1 \)). We discretize the timestamp field into four values: weekday-day, weekday-night, weekend-day, and weekend-night. The time period between 8 AM and 6 PM is considered as daytime.

Table 6.6: Bayesian prediction percent scores with and without timestamp information (without clustering)

<table>
<thead>
<tr>
<th></th>
<th>ECML Data</th>
<th>KDD Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Timestamp</td>
<td>82.94</td>
<td>62.08</td>
</tr>
<tr>
<td>With Timestamp</td>
<td>82.90</td>
<td>61.42</td>
</tr>
</tbody>
</table>

Table 6.7: SVM prediction percent accuracies and percent scores with and without timestamp information (without clustering)

<table>
<thead>
<tr>
<th></th>
<th>Prediction Accuracy (%)</th>
<th>Av. Acc (%)</th>
<th>Score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ECML Data</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Without Timestamp</td>
<td>74.47</td>
<td>46.70</td>
<td>56.73</td>
</tr>
<tr>
<td>With Timestamp</td>
<td>74.35</td>
<td>46.21</td>
<td>56.30</td>
</tr>
<tr>
<td><strong>KDD Data</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Without Timestamp</td>
<td>72.49</td>
<td>20.84</td>
<td>20.59</td>
</tr>
<tr>
<td>With Timestamp</td>
<td>72.69</td>
<td>20.21</td>
<td>20.59</td>
</tr>
</tbody>
</table>

Tables 6.6 and 6.7 give the results of Bayesian and SVM prediction, respectively, with and without time stamp information. For Bayesian prediction, incorporating time stamp information decreases percent scores slightly for both ECML and KDD data. For SVM prediction, incorporating time stamp information decreases percent score slightly for ECML data while percent score increases slightly for KDD data (although average accuracy decreases slightly).

To verify the statistical significance of these results, we apply paired t-test and find that the
increase and/or decrease in percent scores are not significant at 0.05 level of significance. In other words, the null hypothesis that the mean results of without time stamp and with time stamp predictions are equal can not be rejected. From a practical point of view, incorporating time stamp information provides no performance benefit for both of these data sets. Moreover, the efficiency of both models degrades with the incorporation of time stamp information.

**Prediction by Most Relevant Cluster Mode**

The first three page categories visited by a user can be predicted by the mode of the most relevant behavior cluster of the user. These prediction results are summarized in Table 6.8. This approach is simpler and more time and space efficient. Once behavior clustering is done, we use the mode of the cluster that is most relevant to a user for prediction. As such, we avoid estimating and saving thousands of probabilities by compromising slightly on prediction score. Notice that for the ECML data, clustering is able to segment users according to their surfing behaviors as reflected by the increase in prediction scores with the increase in $k$. For the KDD data, the best result is obtained when no clustering is done ($k = 1$). This data set is smaller and less varied, and thus the mode of the entire data is a good representative of all users. These results demonstrate the trade-off between prediction performance and efficiency whereby significant improvement in efficiency can be achieved by a slight reduction in prediction performance.

| Table 6.8: Prediction percent scores obtained when using most relevant cluster mode |
|---------------------------------|------------|------------|------------|------------|
| **ECML Data**                  | $k = 19$   | $k = 10$   | $k = 5$    | $k = 1$    |
| Score (%)                      | **79.40**  | 75.92      | 61.35      | 42.94      |
| **KDD Data**                   | $k = 45$   | $k = 20$   | $k = 10$   | $k = 1$    |
| Score (%)                      | 58.18      | 59.71      | 59.71      | **61.13**  |

**6.5 Conclusion**

In this chapter, we present and evaluate several models for Web surfer behavior prediction studying the impact of behavior clustering on prediction and the effectiveness-efficiency trade-off of various settings. We consider the prediction of the first three page categories visited in a Web navigation
The history of navigation paths is partitioned using a $k$-modes clustering algorithm appropriately modified for the problem of sequence clustering. In particular, we use a new similarity measure called $\text{MatchScore}$ for determining the similarity between two navigation paths and a new approach for selecting initial cluster representatives. Predictions are made by Bayesian and SVM models built on clustered and unclustered data, and by most relevant cluster mode. We evaluate our models on two real world Web surfing data sets.

The following conclusions can be drawn from this evaluation: (1) Behavior clustering can improve surfing prediction using both Bayesian and SVM models. This improvement depends upon the distribution of users across behavior clusters. For our evaluation data sets, which are fair representatives, the improvement in prediction performance is minimal. This can be attributable to the relatively smaller sizes of the data sets. (2) The computational complexity of prediction increases with the number of clusters. (3) The simple prediction approach of recommending the most relevant cluster mode can produce accurate predictions at significantly improved efficiency. (4) Incorporating time stamp information with user ID in the models does not produce significant improvement in prediction. (5) If large quantities of historical data are not available then simpler prediction models or simple predictions based on behavior clusterings are sufficiently accurate for practical purposes.
Chapter 7

Conclusion and Future Work

This chapter concludes the thesis and summarizes our work on text clustering, automatic and autonomic tag recommendation, and Web surfing behavior clustering and prediction. Future works in form of practical applications and research directions are also pointed out.

7.1 Thesis Conclusion

Web mining is an important and practical branch of data mining. Pages of the Web can be considered text documents and mining knowledge out of these documents is a form of Web content mining, which is highly demanded. Text document clustering is a popular task in data mining for understanding and summarizing large document collections such as the Web. Besides the need for efficiency, document clustering methods should produce clusters that are readily understandable as collections of documents belonging to particular contexts or topics. Existing clustering methods often ignore term-document corpus-based semantics while relying upon geometric similarity. In this dissertation, we present an efficient iterative partitional clustering method, CDIM, that maximizes the sum of the discrimination information provided by documents [Hassan and Karim (2012)]. The discrimination information of a document is computed using the discrimination information provided by the terms in it, and term discrimination information is estimated from the currently labeled document collection. Each document is (re)assigned to the cluster for which its discrimination score is maximum. A key advantage of CDIM is that its clusters are describable by their
highly discriminating terms, or equivalently, their highly related terms. We evaluate CDIM both qualitatively and quantitatively on ten text data sets. CDIM can incorporate different information retrieval measures for optimization. We demonstrate clustering using some well known measures including relative risk (RR) [Li et al. (2005)], Measurement of Discrimination Information (MDI) [Cai and Van Rijsbergen (2009)], Domain Relevance (DR) and Domain Consensus (DC) [Navigli and Velardi (2004)]. Repeated bisection versions of CDIM are also presented. In clustering quality evaluation, we find that CDIM produces high-quality clusters comparable to those generated by the best methods. We also illustrate the understandability and efficiency of CDIM, suggesting its suitability for practical document clustering.

We apply our discriminative clustering approach for content-based tag recommendation in social bookmarking systems [Hassan et al. (2009b)]. Our approach uses the novel and efficient discriminative clustering method CDIM that groups posts based on the textual contents of the posts. The method also generates a ranked list of discriminating terms for each cluster. We use the clustering method to build two clustering models one based on the tags assigned to posts and the other based on the content terms of posts. Given a new posting, a ranked list of tags and content terms is determined from the clustering models. The final tag recommendation is based on these ranked lists. If the posters tagging history is available then this is also utilized in the final tag recommendation. The approach is evaluated on data from BibSonomy, a social bookmarking system [BibSonomy (2009)]. Prediction results show that the tag-based clustering model is more accurate than the term based clustering model. Combining the predictions from both models is better than either models predictions. Significant improvement in recommendation is obtained over the baseline method of recommending the most frequent tags for all posts.

We propose and evaluate a self-optimization strategy for our clustering-based tag recommendation system [Hassan et al. (2010)]. To develop this self-optimization strategy for our tag recommendation approach, we empirically investigate when and how to update the tag recommender with minimum human intervention. We present a nonlinear optimization model whose solution yields the clustering parameters that maximize the recommendation accuracy within an administrator specified time window. Evaluation on “BibSonomy” data produces promising results. For example,
by using our self-optimization strategy a 6% increase in average F1 score is achieved when the administrator allows up to 2% drop in average F1 score in the last one thousand recommendations.

Autonomic and autonomous systems exist within a world view of their own. This world view is created from the training data and assumptions that were available at their inception. In most of these systems this world view becomes obsolete over time due to changes in the environment. This brings a level of inaccuracy in the autonomic behavior of the system. When this degradation reaches a certain threshold, self-healing or self-optimizing systems generally recreate the world view using current data and assumptions. However, the self-optimization process is akin to kill a fly with a hammer for minor tuning of the world view. Instead we propose the idea of self-calibration for self-managing these systems [Javed et al. (2012)]. We define self-calibration as the ability of the system to perceive the need for and the ability to execute minimal tuning to bridge the gap between system’s world view and incoming information about the outside world. We present how self-calibration preserves overall system fidelity in our tag recommendation system.

Based on the usage of personomy i.e. a user’s personal tagging history, tag recommendation systems are classified into personalized or non-personalized tag recommenders. A non-personalized tag recommender system does not see a user’s personal tagging behavior while suggesting tags but suggests what is the most popular and relevant. We study the effect of using personal histories in recommending tags [Uddin et al. (2011)]. We experiment on datasets from three popular Web2.0 sites: Bibsonomy, Flickr and Delicious. Three graph based tag recommenders: PITF [Rendle and Schmidt-Thieme (2010)], FolkRank [Hotho et al. (2006)], Adapted PageRank [Hotho et al. (2006)], and our content based tag recommender [Hassan et al. (2009b)] that uses discriminative clustering for recommendations, are applied to study the role of personalization. Important results are obtained in terms of prediction accuracy. We find that users’ preferences for using personal history vary in different datasets which may cause low prediction performance of personalized recommenders. We also present a life cycle of folksonomy users based on the use of their personal history. We propose that this generic life cycle can be exploited by a tag recommender to attain better prediction quality in general across folksonomies.

Web usage mining is a popular branch of Web mining. In this context, we investigate Web
surfer behavior prediction by building generative and discriminative models on the entire history of navigation paths and on behavior clusterings of the history [Hassan and Karim (2011)]. The underlying question that we try to answer is: Does behavior clustering improve behavior prediction? For behavior clustering, we adapt the $k$-modes clustering algorithm [Huang (1998)] by incorporating a new similarity measure that gives greater weight to matches at the beginning of the navigation path. The initial cluster representatives are selected from the set of most dissimilar paths which also fixes the number of clusters. For generative prediction, we adopt Markov chain Bayesian classification models whereas for discriminative prediction we build SVM models. Experiments are performed on two real-world data sets. Surprisingly, the results show that behavior clustering has no significant impact on Web surfer behavior prediction. We also investigate the impact of time of visit, the number of relevant clusters used in prediction models, and the use of cluster modes on Web surfer behavior prediction. We find that for limited scope data simpler approaches such as prediction using cluster modes can produce highly accurate predictions (less than 1% drop from the best prediction) with greater efficiency.

7.2 Future Directions

7.2.1 CDIM Extensions

The evaluation of our discriminative clustering algorithm CDIM suggests that it is a practically useful document clustering method. Its core idea of clustering in spaces defined by corpus-based discrimination or relatedness information holds much potential for future extensions and improvements. An intuitive extension of CDIM is implementing the soft clustering version of CDIM e.g. Soft-CDIM.

Another important direction is investigating different measures of discrimination/relatedness information from text mining and information retrieval domains.

Extending and evaluating CDIM for constraint based clustering is another interesting research direction. Currently CDIM is presented as a uni-clustering algorithm that groups similar documents together and produces a ranked list of terms for clusters meanwhile. The algorithm can be modified...
and a relationship with co-clustering [Dhillon et al. (2003)] can be established such that clusters of documents as well as terms are produced simultaneously. This extension can result in a bi-clustering version of CDIM e.g. Bi-CDIM.

The present CDIM algorithm is an offline algorithm. An interesting and quite practical extension can be an online or incremental version of CDIM.

Incorporation of term semantics while clustering using CDIM is another inviting investigation. Term relatedness weights by WordNet, Wikipedia, Omiotos etc. can be used for semantic normalization of data and its impact on clustering performance can be analyzed.

7.2.2 Tag Recommendation

In the context of tag recommendation systems, one can explore language specific models, incorporation of a tag extractor method, and semantic relatedness and normalization with clustering. Moreover, incremental clustering can be used to reduce the time required for adaptation to drifts and changes. Developing a window based tag recommendation model where less useful information will play a smaller role in recommendation can be worth trying. Producing a fully autonomic tag recommendation system that achieves self optimization through self-calibration is a promising future extension.

In our literature survey of tag recommenders, we also observe that evaluation of tag recommendation algorithm is usually done by comparing F1 score value. No semantic evaluation of recommendations system is normally done except that performing an online user feedback based evaluation. An automated semantic evaluation system can be formulated to evaluate different tag recommendation algorithms.

In real time, a hybrid of content and collaborative filtering technique should be used in recommendation systems. One can analyze different aspects of limitations across datasets for different top notch hybrid recommenders.

So far, we have not analyzed time variations in the history usage of tags. Since we assume history repeats itself and latest tags are more likely to be used than oldest tags, a time aware hybrid tag recommendation system can potentially improve prediction results.
7.2.3 Web surfer behavior prediction

For the Web surfer behavior prediction problem, our investigation can be extended to more and larger Web navigation data sets, and improved hybrid clustering-prediction algorithms for the Web surfer behavior prediction problem can be attempted.
Bibliography

Agarwal A. Web 3.0 concepts explained in plain english. 30.03. 2009, 2009.


Cormode G. and Krishnamurthy B. Key differences between web 1.0 and web 2.0. *First Monday*, 13(6), 2008.


De Nooy W., Mrvar A., and Batagelj V. *Exploratory social network analysis with Pajek*, volume 34. Cambridge Univ Pr, 2011.


Dempster Arthur P., Laird Nan M., and Rubin Donald B. Maximum likelihood from incomplete


Kriegel H.P., Kröger P., and Zimek A. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1, 2009b.


Li H., Li J., Wong L., Feng M., and Tan Y. P. Relative risk and odds ratio: a data mining per-


Pantel P. and Lin D. Discovering word senses from text. In *KDD ’02: Proceedings of the eighth*


