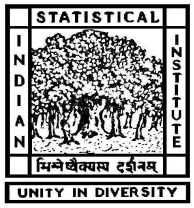
WEB SURFER MODELS: PREPROCESSING, PAGE RANKING, AND QUANTITATIVE EVALUATION

Thesis Submitted to
INDIAN STATISTICAL INSTITUTE



2008

by

NARAYAN L. BHAMIDIPATI

Machine Intelligence Unit, Indian Statistical Institute
203 B. T. Road, Kolkata, India

To my mother

Declaration

The work in this thesis is based on research carried out at the Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgments

It is a great privilege and pleasure to have worked with Prof. Sankar K. Pal, who was not just a thesis supervisor, but a mentor, teacher, and friend, too. His methodical approach, attention to detail, academic achievements, administrative capabilities, and international recognition are all inspiring, as well as, awe-inspiring.

I am extremely grateful, and forever indebted, to Prof. C. A. Murthy for having advised me to pursue research as a career — perhaps, the most timely piece of advice that I had ever received (and heeded). He has always been a great asset to the Institute, and, in particular, of great help to me, thanks to his mathematical and statistical abilities combined with his knowledge of computer science subjects.

Heartfelt thanks to the omniscient Dr. Pabitra Mitra, and the ever-so-approachable Dr. Mandar Mitra, for having advised, inspired, and helped me in more ways than one, and for having introduced me to a variety of subjects and concepts during the past few years.

I express my gratitude to Prof. Palash Sarkar for being a wonderful teacher who went out of his way to introduce me (and my classmates) to C Programming – perhaps, my first stepping stone towards this thesis and career. My old friend, Deepayan Sarkar, too, deserves my gratitude for having introduced me to the world of fractals and computer graphics (on VT320 terminals for VAX/VMS), apart from gifting me my first Knoppix Live CD.

I gratefully acknowledge INSEAD, France, for having offered me its prestigious fellowship (including funding my research at a time when I was financially weakest). Prof. Soumitra Dutta, Dean at INSEAD, France, apart from being a co-author, has always been

an amicable person I could turn to for advice and suggestions.

I am indeed indebted to Yahoo!, and the Data Mining & Research team, for giving me not only my first job, but also a great career and a wonderful work environment. Specifically, the credit goes to Jignashu Parikh, Dr. Rushi Bhatt, Dr. Rajesh Parekh, Dr. Nicolas Mayoraz, Dr. Vijay Narayanan, and Dr. Pavel Berkhin, for having mentored, trained and encouraged me both on the work and personal fronts, and for also discussing and providing valuable feedback regarding my research and thesis on many occasions in the past couple of years.

I acknowledge the immense favors I got from the WebBase group at Stanford University when I was still new to the subject of Web Mining, and needed standard data sets for my experiments. In particular, Wang Lam, Sriram Raghavan, Gary Wesley, and Taher Haveliwala had spent time and effort helping me obtain pages and code from WebBase, and understand the intricacies involved. Also, volunteers who spared their valuable time to help me in my experiments are being thankfully acknowledged.

Sincere thanks to all the faculty, students and office staff of the Machine Intelligence Unit and the Center for Soft Computing Research, each one of whom made my student life absolutely memorable. Among them, Dr. B. Uma Shankar, Prof. Malay K. Kundu, Prof. Ashish Ghosh, Prof. Sushmita Mitra, Dr. Sambhunath Biswas, Dr. Deba Prasad Mandal, Dr. Rajat De, Dr. Sanghamitra Bandyopadhyay, Dr. Pradipta Maji, Dr. Kuntal Ghosh, Dr. Sarif Naik, Praveen Tripathi, Ms. Minakshi Banerjee, Shubhra Sankar Ray, Suman Saha, Debasis Sen, and Indranil Dutta, deserve a special mention.

Special thanks to Dr. Susmita Sur-Kolay, Prof. Bhabatosh Chanda, and Late Prof. P. K. Nandi for having served on my annual evaluation committee on multiple occasions and providing suggestions for enhancing my work.

Prof. S. C. Bagchi, Dr. Arnab Chakraborty, Dr. Utpal Garain, Dr. Saurabh Ghosh, Prof. Alok Goswami, Dr. Subhamoy Maitra, Dr. Krishanu Maulik, Prof. Partha Pratim Mazumder, Prof. Dipti Prasad Mukherjee, Dr. Krishnendu Mukhopadhyay, Dr. Amita Pal, Prof. N. R. Pal, Prof. Bimal Roy, and Prof. K. K. Roy have all been great teachers

and friendly people at the same time.

I am indeed grateful to our friendly, neighborhood, unix administrator, Subhasis Pal, and the staff of the Library and Dean's Office for having helped me on various occasions.

Anshul, Sunil, Arvind, Ravi, Krishna, Apoorva, Rik, Sarang, Anindya Sen, Prem Laxman Das, Vikal Tripathy, Avishek Adhikary, Somitra Sanadhya, Naveen Jana, Joydeep Jana, Durga Prasad Muni and Lingaraj Sahu, have all been great friends, and some of them also helped me academically through their thoughtful comments and discussions.

Last but not the least, I want to thank the whole of my extended family and the friendly Sundarams and Chatterjees for being with me throughout. Prof. Y. R. Sarma, Prof. B. V. Rao, Late Prof. A. R. Rao, Prof. S. B. Rao, Prof. T. J. Rao, Mr. C. H. Sastry, Prof. Joseph Mathew, Dr. T. S. Vasulu, and Dr. Arni Srinivasa Rao were not just faculty or teachers, but, along with their families, they have been (and still are) friends and well-wishers for both me and my family — to an extent that I can consider them as part of my family. I owe many of them all that I learnt in life, especially, in the latter part of it.

This section cannot end without acknowledging Linux (in the form of a Kubuntu Live CD), which has recently assisted me in copying out all the source files of a version of this thesis undergoing final touches (which was not backed up for a couple of days) after Windows locked me out of my laptop.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Scope of the Thesis

1.1 Introduction

The World Wide Web [12] (usually referred to as the *Web*, *WWW* or *W3*) is an enormous collection of data available over the *Internet*, which is a vast network of computers. It was created in the year 1990 by Tim Berners-Lee, while he worked at CERN, Switzerland, and was made available over the Internet in 1991. The World Wide Web Consortium [136] authoritatively defines the Web as "the universe of network-accessible information, the embodiment of human knowledge". The Web consists of objects, also called documents or pages in a generic sense, that are identified using a Uniform Resource Identifier (URI), or what has more popularly come to be known as the Uniform Resource Locator (URL), and these objects are connected to each other by means of *hyperlinks*. This interlinked nature of the Web distinguishes it from text corpora and other such collections.

The Web is a rapidly changing and expanding resource, and over the years, it has seen a phenomenal growth in both its size and diversity. Starting from a single web site (`info.cern.ch`) in 1990, it is now made up of millions of web sites. Currently, the indexable Web itself consists of billions of heterogeneous documents — in the year 2005, Yahoo! [145] had announced to have indexed *over 20 billion* documents [146], and Google [52] countered this statement by claiming to have indexed *at least thrice as many*

documents than that [53]. These are all under-estimates, as they take into account only what has been crawled, and the true size of the Web is *unknown*.

There are a wide variety of data sources that contribute to the richness of the Web. We list a few of these so that one may gauge the root of the heterogeneous nature of the Web:

- Content created for the Web, and published by the authors: This is usually made up of textual (in either text or HTML formats), image, audio, or video content, and is generally created and distributed by professionals.

- Content created for other purposes, and now made available on the Web: Examples include music, movies, and printed books.

- Public mailing lists and discussion forums: A lot of knowledge as well as entertaining articles are shared in this form.

- User generated content: This recent phenomenon is about content being generated by end-users, as opposed to professionals, and is changing the very face of the web. This includes images, audio and video submitted by users of web sites and blogs.

- The deep web: This refers to content that is generated on the fly, and is generally not indexed by search engines. New content that is being generated may be in response to some user input (explicit inputs may be user's identity or query terms, while implicit input could be geographic location of the user's IP or browsing context), or may depend on other factors (examples include time and changes in other parts of the Web).

- Activity logs: While the earlier data sources were explicitly created by authors and visitors of web sites, their activity itself, when recorded and stored, constitutes another type of web data. In a sense, every action on the web, be it viewing a web page, writing a mail, or uploading an image, to name a few, generates new content. Of course, not all of this may be stored for a long period or be made publicly available.

The size and heterogeneity of the Web present immense challenges for knowledge discovery. Knowledge discovery in databases (KDD) [49] is aimed at discovering natural and interesting structures within such massive and often heterogeneous data. KDD stands on the shoulders of the giant literature in *Pattern Recognition* ([41,45,51,116], to name a few) which predates the existence and availability of massive databases. However, KDD is being visualized as not just being capable of knowledge discovery using generalizations and magnifications of existing and new pattern recognition algorithms, but also the adaptation of these algorithms to enable them to process such data, the storage and accessing of the data, its preprocessing and cleaning, interpretation, visualization and application of the results, and the modeling and support of the overall human-machine interaction.

Data mining [56, 114] is that part of knowledge discovery which deals with the process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data, and excludes the knowledge interpretation part of KDD. Data mining refers to data in a general sense, and the basic techniques are applicable to various domains such as text, web, or biological data, where one may make use of additional domain-specific knowledge. *Web mining* is data mining for the web domain, and is defined as the extraction of interesting and potentially useful patterns and knowledge from objects or activity on the Web.

Web mining tasks include page summarization, ranking, categorization and clustering, user modeling, and personalization. Among these, page ranking is one of the most important tasks, whereby each web page is assigned a score reflecting something like the popularity or authority of the page. Most present day page ranking algorithms are variants/combinations of the earliest page ranking algorithms, HITS and PageRank, which were both developed around 1998 and had adapted bibliometric and sociometric ideas to the Web. Interestingly, both HITS and PageRank model the Web as a directed graph, and fall under the category of link analysis algorithms. Moreover, both HITS and PageRank algorithms may be interpreted as surfer models which study the long term behavior of a web surfer under various assumptions of traversing the set of available web pages.

The objective of this thesis is to present the results of some investigations, both theoretical and experimental, addressing certain tasks essential for surfer modeling, page ranking, and web mining, in general. Tasks considered include preprocessing text and links in web document collections, providing new and better surfer models, and comparing ranking algorithms. Before we describe the scope of the thesis, we provide a brief review of web mining, surfer models, and page ranking, a discussion on some challenges involved, and possible solutions.

Section 1.2 presents a brief overview of web mining and page ranking. Section 1.3 discusses the Web in a graph theoretic framework. We then study web surfer models in Section 1.4. Preprocessing web data and comparing ranking algorithms are discussed in Sections 1.5 and 1.6, respectively. The scope of the thesis is presented in Section 1.7.

1.2 Web Mining and Page Ranking

Web mining deals with the application of data mining techniques on data available from the Web. There are roughly three knowledge discovery domains that pertain to web mining: Web Content Mining, Web Structure Mining, and Web Usage Mining [26, 76, 91].

Web content mining [76, 91] is the process of extracting knowledge from the content of documents or their descriptions. The heterogeneous and semi-structured nature of the ever expanding information sources on the Web makes automated discovery, organization, and management of Web-based information difficult. Content mining tasks include page classification, clustering, summarization and relevance computation.

Web structure mining [26, 76] is the process of inferring knowledge from the interconnections of the Web documents induced by the hyperlinks between them. The existence of link-based information distinguishes Web document collections from text corpora, as hyperlinks induce relations between the linked documents. Page ranking and detecting communities are two of the most common structure mining tasks. Web content mining and web structure mining, when used together, produce powerful methods of analyzing

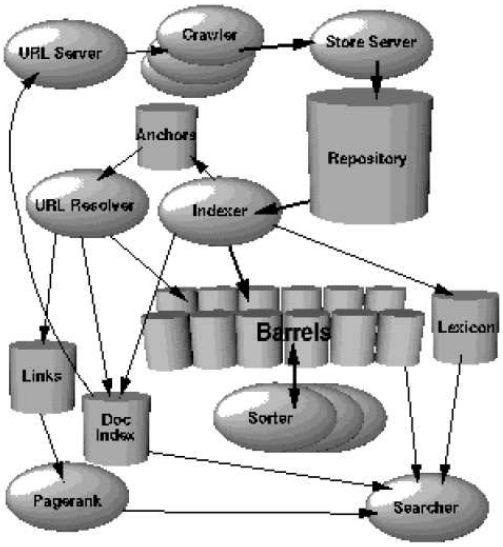


Figure 1.1: Anatomy of a large-scale hypertext search engine [22]

web data. Page ranking, classification and clustering are all prime examples of tasks that benefit from the marriage of web content and structure mining.

Web usage mining [33, 91] attempts to discover useful knowledge from the secondary data obtained from the interactions of the users with the Web. As users browse the Web, foraging for information, they leave behind valuable information in terms of their online behavior. This information may be utilized to improve the capability of the servers to better satisfy their users. For example, understanding user behavior may help in site re-organization and personalization.

We now elaborate upon page ranking which is one of the most important and complex web mining tasks. When a user searches the Web with a query, a lot of pages may contain or match the query, but only a few would be interesting and relevant to the user. Obtaining scores to be used for ordering these web pages and placing the most relevant results at the top, is known as page ranking. Fig. 1.1 shows where page ranking fits in the architecture of search engines [22].

Ordering on keyword based relevance has been widely studied in text mining and may be easily extended to the Web. However, new challenges appear in the web scenario,

because unlike text collections, the Web may contain a lot of spam. Web page authors may deliberately insert irrelevant or catchy keywords into their content, in an effort to entice search engines to regard their pages as more relevant for certain queries. Thus, page ranking has to consider other measures such as popularity, authority, trust, *etc.*along with relevance so that the "best" results appear at the top.

Several notions of popularity or authority are available in the literature, and are based on the link structure of the Web, where a link is assumed to be a vote by the originating page in favor of the destination page. The indegree algorithm [134] rated a page highly if several pages pointed to it. Again, this algorithm may be easily manipulated by creating a network of pages, with each page pointing to all the rest. Brin and Page [22] came up with a normalized version of this algorithm whereby each page could obtain a certain number of votes from its predecessors, and redistribute it equally into all its successors. This simple algorithm, called PageRank, performs well enough to be used by the biggest search engines.

1.3 The Web as a Graph

Both Web content mining and Web structure mining may be studied simultaneously by treating the Web as a directed graph, with the documents forming the vertices and the hyperlinks between them considered to be the arcs of the digraph. Broder *et al* [23] had studied a huge set of web pages crawled by the AltaVista search engine, and found some surprising properties regarding the structure of the Web. Their crawl had consisted of about 200 million pages, and 1.5 billion links. The pages were divided into roughly four equal parts:

- A Strongly Connected Component (SCC), where all the pages are reachable from each other.

- IN, containing pages that lead to the SCC but cannot be reached from it.

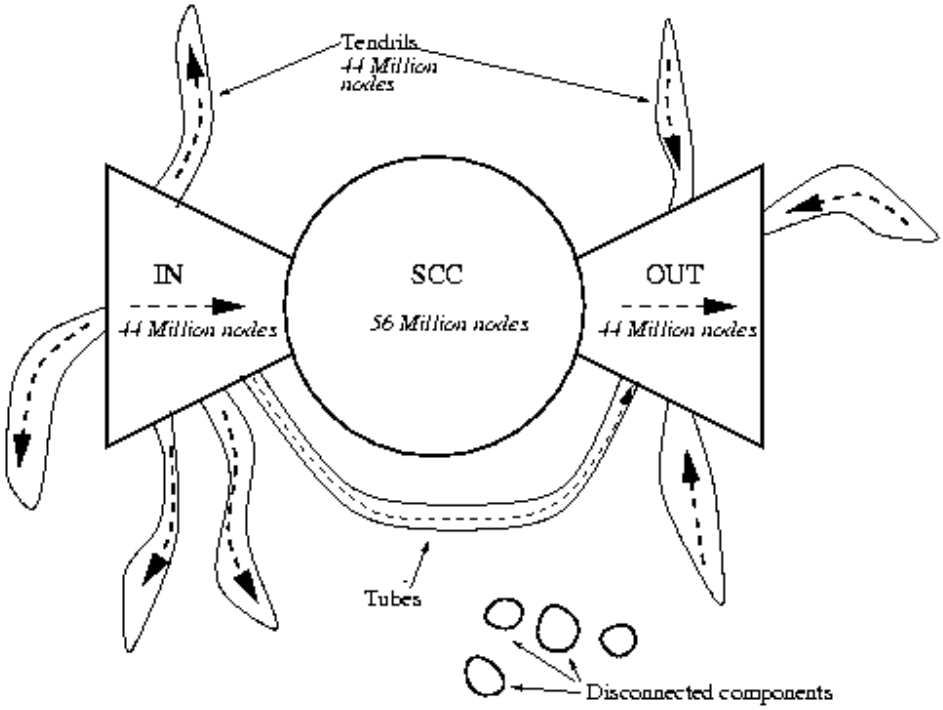


Figure 1.2: Bow-tie structure of the web digraph [23]

- OUT, containing pages that can be reached from the SCC but do not lead back to it.
- Tendrils, which consists of pages that can neither be reached from the SCC nor lead to the SCC.

Apart from these, there are some small disconnected components. Fig. 1.2 shows the structure of the Web as depicted in [23], which eventually came to be known as the *bow tie structure*.

Modeling the Web as a graph helps us utlize the abundant graph theoretic literature for understanding various properties of the Web. Interesting properties of the graph are extracted by studying the access patterns of the nodes by real users. For example, if a page is accessed more frequently than another page, the first page may be called more popular than the second one. Often, instead of observing the user directly, it is easier to create a model of her surfing behavior. These models, known as surfer models, assist in educing

useful information from the given web data. The next section provides a brief description of surfer models, their theoretical background, some existing models, and applications.

1.4 Web Surfer Models

Surfer models model a user who browses the Web by considering the surfer's behavior to be a random walk on the web graph. The surfer is assumed to be clicking on links or typing URLs to move on to new pages. With time, the state of the surfer keeps changing, and this behavior may be treated as a stochastic process. At each time point (t), a snapshot of her behavior, which is the state she is in at t, is available. This process is, generally, assumed to satisfy the Markov property, which means that the present behavior of the surfer does not depend on past history (i.e., beyond a certain point back in time). Also, it is assumed that the behavior of the surfer is time-invariant, i.e., given her current state, the same behavior would be expected irrespective of the time at which this state has been achieved.

A brief note about the notion of time is in order here. When modeling the surfing behavior of a user, the concept of time (t) may be considered in two different ways. One way is to treat the total time since the surfer has started surfing as the time t, resulting in a continuous stochastic process. The other is to consider the number of pages (including repetitions) the surfer has traversed before reaching the current page, and thus, the time is discrete. Generally, it is the second one that is more in use, perhaps due to the ease of observing it through the explicit actions (clicks) of the surfer.

This discrete sequence of states being traversed by the surfer, is a random walk on the web graph, and gives rise to a stochastic process $\{X_t\}$ which denotes the state the surfer is on at time t. The state of the user may be defined, as the situation demands, to be the web page she is currently browsing, her topic of interest at that moment, *etc.*. The objective is to study this model from various angles, find interesting properties, and make appropriate interpretations about web users without involving real users directly.

This serves as an alternative to observing real surfers which is a rather complex and time consuming task, and involves issues like preserving privacy of individual users. Moreover, such information may turn out to be noisy or biased and may not be applicable to an average or particular individual browsing the web.

These models make extensive use of Markov chain theory, which we briefly describe now.

1.4.1 Markov Chain Theory

In this section, we review some elementary properties of Markov chains which would be useful from the point of view of surfer models. For a more detailed treatment of the subject of Markov chains, one may refer to [67].

A Markov chain is a sequence of random variables $X_0, X_1, X_2, \ldots$, satisfying the Markov property, namely, given the present state, the future and past states are independent. The set of all states that the random variables may assume is called the *state space*, and is denoted by S. In the whole of this thesis, the state space being considered is finite. Also, the chain is assumed to be time-invariant, meaning that the random variable $X_{n+1}|X_n$ is independent of n. This is also known as the memorylessness property. Thus, the probabilities for a transition from a state i to state j may be specified without referring to the time points. A matrix whose (i, j)th entry corresponds to the probability of a transition from the ith state to jth state is called the transition probability matrix. This matrix determines the Markov chain, and thereby, all its properties.

A Markov chain is irreducible if any state is reachable from any other state. A set of states is called closed if no state outside it can be reached from a state within it. For an irreducible Markov chain, no proper subset of S is closed. A state is recurrent if the chain would surely return to that state again in finite time. All states of a finite irreducible Markov chain are recurrent.

A recurrent state is called periodic if it is impossible to return to the state except at regular intervals. A state is aperiodic if it is not periodic, in which case, beyond a

sufficiently long period of time, it is possible to be in that state for any time point. A Markov chain is aperiodic if all its states are so. If one state of an irreducible Markov chain is aperiodic, then so are all the other states, and thus the chain itself is aperiodic.

A finite, irreducible and aperiodic Markov chain has a unique stationary distribution, meaning that the probability of the chain being in a given state converges with time to a unique number. The convergence is a consequence of aperiodicity (for the probabilities would have been oscillating if a state were periodic), whereas, the uniqueness results from irreducibility, and thereby, whichever state the chain starts from, it would always converge to the same distribution. The stationary distribution represents the long term behavior of the chain and smooths out any initial biases or preferences.

We shall also describe Fuzzy Markov chains which have been studied in the present thesis as a robust alternative to the classical Markov chains based on probabilistic transition matrices. Fuzzy Markov chains are similar to the classical Markov chains but operate on the fuzzy algebra instead of the classical algebra. Whereas we have the usual addition and multiplication on the classical algebra, these operations are changed to $\max$ and $\min$, respectively, in the fuzzy algebra. That is why fuzzy algebra is also known as $\max$-$\min$ algebra.

The fuzzy counterparts of transition matrix, stationary distribution, *etc*.are defined similar to those in the classical case [4]. For example, in the fuzzy case, the transition matrix contains fuzzy numbers [4], instead of probabilities. The (i, j)th element of this matrix denotes the belief (as opposed to the probability) of making a transition to state j when on state i. It is known that minor changes in the probabilistic transition matrices may result in big differences in the limiting distributions, whereas fuzzy Markov chains are more robust to changes in the entries of the transition matrix [4].

While classical Markov chains have been in use since being introduced in 1906 by A. A. Markov [10], Fuzzy Markov chains are relatively recent, first appearing in [79]. While there is a rich literature dealing with the various properties of fuzzy Markov chains, a few properties regarding counterparts of classical Markov chains still remain unknown. For

example, the conditions for regularity of fuzzy Markov chains still remain elusive.

We now elaborate on how surfer models may be viewed as Markov chains.

1.4.2 Surfer Models as Markov Chains

In order to cast surfer models as Markov chains, we would need to make the correspondences between the two definitions. The state of the chain is simply the state of the user. However, as mentioned earlier in Section 1.4, the definition of the state of the user may be varied to accommodate whatever features of the user one is trying to study.

Once the state space is fixed, all that needs to be done is to define a transition matrix. This is a crucial step where one tries to incorporate all available domain knowledge about surfing patterns with the objective of providing a model very close to reality. The (i, j)th entry of the transition matrix is determined by what is deemed to be the probability or belief value of moving on to the jth web page given that the surfer is presently on the ith page. The computation of this probability may take into consideration several factors like the content and link structure of these two and other pages, the total number of pages available, topics of interest, preferences of users, and any other assumptions being made on the movement of the surfer.

Now that one can cast the surfer model as a Markov chain, there has to be a system of interpreting the results obtained in terms of the properties of the chain. A visit of the chain to a particular state may be thought of as a visit of the surfer onto the corresponding web document. Similar interpretations may be made regarding the frequency of visits, especially, after the chain has run for a long time. The stationary distribution of the chain assigns a probability value to each state, and these values can be thought of as the (unconditional, time-independent) probability of a surfer being on the corresponding states.

In the case where the surfing pattern is being modeled as a classical Markov chain, for the stationary distribution to exist and be unique, one has to ensure irreducibility and aperiodicity. Surfer models can achieve aperiodicity easily. If a surfer is allowed to stay

on the same state (i.e., the transition does not change her state), then the probability of being on that state at any time point (after a sufficiently long time) is non-zero. Of course, for the previous statement to hold regardless of the initial state of the surfer, it is being assumed that the chain is irreducible.

Guaranteeing irreducibility of the chain is not as straightforward as ensuring aperiodicity. The questions that need to be taken care of are as follows:

- What does a surfer do when she reaches a node with no outgoing arcs?
- How can she be sure of reaching each state in the state space? In other words, how can she be sure of not getting trapped in a closed subset of states?

Nodes with no outgoing arcs are called dangling nodes. When a surfer lands on a dangling node, she is allowed to move on to any of the available nodes. This means that artificial arcs are added to dangling nodes (and they are no longer dangling). Interestingly, if the surfer is allowed to jump from any node to any other node, irreducibility is assured. Intuitively, this indicates that the surfer may always chose, with some probability, to jump to a new random URL instead of following one of the outlinks.

Thus, we see that, specifying a surfer model involves deciding upon how the transitions are allowed to take place, subject to the constraints of irreducibility and aperiodicity. We now describe some such existing models.

1.4.3 Existing Models

Several surfer models have been introduced over the past decade, of which the random surfer model is widely used. In the random surfer model, the surfer is assumed to be moving on to new pages at random by clicking on the links of the current page, occasionally choosing to jump to a new page. This extremely simplistic and easy-to-interpret model captures what is meant by the popularity of a page, and is at the core of the immensely popular PageRank algorithm.

Another surfer model that has spawned several variants is the *directed surfer model* [123]. The random surfer model assumes that the surfer is browsing web pages at random by either following a link from the current page chosen uniformly at random or by typing its URL. On the contrary, the directed surfer model assumes that, when the surfer is at any page, she jumps to only one of those pages that are relevant to the context, the probability of which is proportional to the relevance of each outlink. Both models guarantee the convergence of this stochastic process to a stationary distribution under mild assumptions like the irreducibility of the transition probability matrix. In practice, these assumptions are enforced by pruning or ignoring some links.

SALSA [86] presents a surfer model interpretation of the HITS algorithm [73], and involves transforming the web graph into a bipartite graph. Here the (i, j)th element of the transition matrix is defined to be the probability of reaching j from i by going, at random, to one of the pages linking to i and then choosing one of the links available on that page, again, at random.

1.5 Preprocessing Web Data

Preprocessing is an important step for mining tasks, whereby, the features of a data set are modified so as to make information extraction reliable and convenient. Preprocessing is necessitated due to one or more of the following reasons [132]:

- presence of noise in data: noise may disturb the information extraction process by making the data less than ideal.
- sparsity of data: this results in a lack of information regarding certain portions of the data space, and consequently, inference cannot be generalized easily to unseen examples.

For web data, there are two aspects in this regard: text preprocessing and link preprocessing.

1.5.1 Text based Preprocessing

Textual content may be noisy due to various reasons [139], some of them being:

- Words spelt wrongly or in an alternate way
- Presence of synonyms, homonyms, *etc.*
- Presence of stopwords or irrelevant words
- Use of several related but different words
- Presence of unexpected or foreign words
- Improperly formed or unclosed tags

Each one needs to be treated in its own way as they would otherwise interfere in basic tasks such as tokenizing, indexing, and retrieval. Misspelt words and word variants may be detected using a pre-defined dictionary, and corrected using either a table lookup, or choosing valid words with a small edit distance from these words [80]. Stopwords may be removed by using an exclusion list [126], though in some studies, they are retained because of the value they provide in terms of context [89]. Similarly, some studies (e.g., [125]) ignore HTML tags for the sake of simplicity, while others (e.g., [87]) retain them because of the richness they provide to the textual contents. Taking care of related words is probably the most difficult and challenging part of preprocessing text data. Part of the challenge lies in defining what “related” means, and the other part is to find groups of words which can be clubbed together.

1.5.2 Link based Preprocessing

Link preprocessing [91] is an extremely important step for link analysis algorithms. For example, ignoring all links between pages in the same domain would yield vastly different results than what one would have obtained when all those links were included. Detecting

and removing noise introduced by hyperlinks is a more complex task than the corresponding operation for textual content. Apart from the presence of *dead hyperlinks* (that is, links to non-existent web pages), links to *irrelevant content* is a major cause of web page noise. Judging which links lead to irrelevant content is a difficult job because of the variety of roles that hyperlinks play. Also, the very notion of relevance may change over time, one of the reasons being that the target page has changed since the link was created. Apart from noisy links, there are links which are deliberately introduced for spamming search engines. Detecting and fighting link spam is a major field of study on its own.

1.6 Quantitative Comparison of Score based Ranking Schemes

Once we have competing surfer models or page rank algorithms that produce score vectors for the set of available documents, one may want to find how different the alternatives are. For this particular case, there are two possible ways of comparison, namely, score based and rank based [122]. Score based methods compare the underlying scores directly, without considering the impact on ranking. Rank based methods first compute the rankings induced by the scores, and then compare only the rankings, while ignoring the scores totally.

1.6.1 Score Based Comparison

Score based comparison is usually performed by computing either the dissimilarity or the similarity between the two vectors. Measures such as the Minkowski distance of order p (popular choices of p being 1, 2, and ∞, which result in the Manhattan, Euclidean and Chebyshev distances, respectively) measure the dissimilarity, whereas, correlation and the cosine of the two vectors measure similarity between them.

1.6.2 Rank Based Comparison

Rank based comparison, on the other hand, is performed by first converting the scores into the corresponding rankings and then computing similarity or dissimilarity between the rank vectors. Once again, the Minkowski distances or the usual correlation measures may be computed on the rank vectors, resulting in measures such as Spearman's footrule and Spearman's correlation. Moreover, one may also make use of the concepts of concordance and discordance [32] to compute the Kendall distance (also known as Kemeny distance or bubble sort distance) between the two rank vectors.

As noted earlier, existing score based, as well as, rank based comparison methods work in isolation, and either neglect the ranking perspective, or ignore the additional information contained in scores.

1.6.3 Rank Fusion and Score Fusion

Given multiple sets of ranks or scores for the web documents under consideration, fusion is the process of combining them to obtain a single set of ranks or scores. Rank fusion [100, 130], also known as rank aggregation [46, 151], obtains a consensus ranking from the available ranked lists. These lists need not be full lists, making rank fusion a very challenging problem. Score fusion, on the other hand, combines the scores directly, in order to produce a consensus score vector, on which the final ranking may be based upon. Two of the standard score fusion techniques are CombSUM (a simple average) and CombMNZ (a weighted average) [83, 122].

1.7 Organization and Scope of the Thesis

The present thesis provides some new results of investigation, both theoretical and experimental, concerning certain tasks related to Web surfer modeling. The tasks include text and link preprocessing, page ranking, page categorization and quantitative evalua-

tion. Methodologies developed are based on both classical probability theory and fuzzy logic, to model surfing patterns, and they provide a strong mathematical framework for comparing them on the basis of the resultant page ranks.

Our contribution to text preprocessing is the development of a novel corpus-based stemmer while, that for link-preprocessing involves detecting relevant sequences and cycles of web pages in the web graph. Novelty in the two surfer models developed in the thesis is as follows: A topic-continuity based web surfer model is mathematically formulated to incorporate the tendency of users to continue browsing on a particular topic. In another model, the notion of fuzzy hyperlinks is introduced to develop a fuzzy web surfer model based on the theory of fuzzy Markov chains. Apart from studying the properties of surfer models, a new metric has been proposed for measuring how different the resultant page rank vectors are. This metric computes distance between two rankings directly on the basis of the underlying score values, and generalizes Kendall distance. Also, a top k version of the metric is obtained which is particularly useful in comparing page ranks when the number of documents is very large. The effectiveness of the methodologies described in the present thesis is demonstrated on various data sets used in the context of text mining and web mining. Superiority of the models over related ones is established statistically.

The thesis consists of five contributory chapters apart from the Introduction (Chapter 1) and the Conclusions, Discussion and Scope for Further Works (Chapter 7) chapters. A chapter-wise summary of the thesis is provided below.

Chapter 2 STEMMING FOR TEXT PREPROCESSING OF WEB DATA [14, 103]

Preprocessing is an important step for mining tasks, whereby, the features of a data set are modified so as to make information extraction reliable and convenient. For web data, there are two aspects in this regard: text preprocessing and link preprocessing. Among the text preprocessing tasks, stemming, whereby semantically similar words are grouped together, is an important and often used technique. Stemming has been generally observed to improve recall in information retrieval. However, there is no agreement on the same for

the case of classification. We describe a novel corpus-based stemming technique which models the given words as being generated from a multinomial distribution over the topics available in the corpus. A sequential hypothesis testing like procedure helps us group together distributionally similar words. This stemmer refines a given stemmer and its strength can be controlled with the help of two thresholds. We have tested the proposed methodology on three data sets and found that, despite a huge reduction in dictionary size, the classification accuracies and retrieval precision have significantly improved in most cases. It has also been found suitable for cross-corpus stemming. We also evaluated this stemmer linguistically on the basis of error counting methods and found that even without any prior knowledge of the language specific properties of the words, the stemmer performs remarkably well.

Chapter 3 SEQUENCE DETECTION FOR LINK PREPROCESSING OF WEB DATA [102]

Link based preprocessing is necessitated by the fact that useful structural information on the web is often accompanied by a large amount of noise such as banner advertisements, navigation bars, copyright and privacy notices, etc. Such items often hamper automated information gathering and web data mining tasks like web page clustering, classification, information retrieval and information extraction. Link cleaning leads to performance improvement for the above mentioned tasks.

The same content on the web may be present in a single document or may be split into several parts. Since the problem of page ranking is all about comparing documents competing with each other in terms of their content and link structure, this leads to the question of fair comparison.

In order to deal with such discrimination, we develop efficient and scalable algorithms to detect content that could have been merged but has been spread over several documents just for the sake of convenience or presentation. This involves detecting special graph structures in the web graph, like sequences of web documents terminating in a leaf node, or a cycle of web pages. These algorithms are based on a simplistic notion of finding "next" and "previous" elements of a sequence by looking at the relations between them,

as reflected by the position of the links, the amount of surrounding text, and other such features. This kind of link preprocessing not only eliminates several mirror pages, which would not have been detected by the existing algorithms, but also leads to the novel idea of returning sets of pages as results from search engines.

Chapter 4 WEB SURFER MODEL INCORPORATING TOPIC CONTINUITY [101, 115]

Web surfer models study various aspects of web mining by realistically modeling web users. Since the objective throughout is to maximize the gain of the end user (or supply sufficient information to a service provider, who in turn, may pass the benefit to the end user), appropriate web surfer models for real users are very useful. Once a model is known to be reasonable, one does not need to track real surfers, and can simulate the user's behavior directly from the model, thus, saving valuable resources like time and money, while maintaining privacy.

Surfer models simulate the behavior of web surfers by modeling the sequence of pages visited as a stochastic process and extract useful and interesting information about the web. In particular, they can be used to compute the ranks of a web page as the unconditional probability of a surfer being on that page under the assumed model. It has been observed that the use of context information improves page ranking [123]. In particular, the continuity of topics that a surfer would maintain while browsing the web would provide valuable information about the transition probabilities of the model.

In this chapter, we describe a web surfer model that incorporates the notion of topic continuity. Therefore, unlike earlier models, it captures the inter-relationship between categorization (context) and ranking of web documents simultaneously. The model is mathematically formulated. A scalable and convergent iterative procedure is provided for its implementation. Its different characteristic features, as obtained from the joint probability matrix, and their significance in web intelligence are mentioned. Both theoretical and experimental results confirm the superiority of the model. Experiments are performed on web pages obtained from WebBase.

Chapter 5 WEB SURFER MODELS INCORPORATING FUZZINESS [15, 104]

In the previous chapter, surfer models were studied where the uncertainty in the surfer's transition from one page to another was modeled probabilistically. In some cases, probabilistic models fail to capture the inherent variety of uncertainty. In this chapter, we demonstrate the need for fuzzy web surfer models through some examples. In particular, we deal with fuzziness in links between pages, especially, when links that are intended to point to particular sections of web pages do not do that explicitly.

A novel web surfer model is introduced where the transitions between web pages are fuzzy quantities, and FuzzRank is defined, similar to PageRank, as the principal fuzzy eigenvector of the fuzzy transition matrix. Whereas, the usual web surfer models are based on the theory of Markov chains, the proposed model is based the on theory of fuzzy Markov chains. In this manner, besides being able to model the inherent fuzziness in links and contexts, the model inherits the advantages of fuzzy Markov chains, namely, finite convergence, and robust computation. Also, a study is conducted into the ergodicity properties of fuzzy Markov chains, and the efficient computation of FuzzRank. Experiments performed on data sets from WebBase support the theory regarding the stability of fuzzy surfer models.

Chapter 6 QUANTITATIVE EVALUATION OF PAGE RANKING SCHEMES [13, 16]

In Chapters 4 and 5, we have described and studied various page ranking schemes. While, it is clear that they produce different page rank values, two questions emerge immediately:

- Which scheme is the best scheme? This involves comparing pairs of schemes.
- How distinct are the schemes in terms of the ranks they produce? Are they significantly different?

The former has been widely studied in the context of page ranking, where most of the approaches are subjective comparisons made between the competing schemes. The latter is a relatively ignored problem, with the two page rank vectors compared only in terms of the rankings produced by them.

This chapter is concerned with quantifying how much page ranking schemes differ from each other. To this end, a generalized Kendall distance, which can compare more than just page ranking schemes, is defined. Metric properties for this newly introduced measure are proved. The generalized Kendall distance looks at not only the final ordering that the two schemes produce, but also at the spacing between pairs of scores. We take a fusion based approach, whereby, two rank vectors are the same as each other if they produce the same ranking on fusing together with another score vector (say, relevance to a query).

A parameter γ in the definition of the metric, takes into consideration the potential uses of the score vectors to be compared. It is shown that the classical Kendall distance may be obtained as a limiting value of our metric, as $\gamma \to \infty$.

A top k version of the metric is provided, which replaces the unknown values by their expected values. The detailed computations are provided in the Appendix. Several mathematical properties of the newly introduced metric are stated and proved.

Applications include comparing two page rank vectors, deciding on the stopping time of the power iterations and measuring how well the ranks represent the scores. In particular, since the Kendall distance between a score vector and corresponding rank vector is always zero, the last application sets our methodology apart from the usual rank comparison methods. Experimental results on both small and large data sets depict the utility of this new metric.

Chapter 7 CONCLUSIONS, DISCUSSION AND SCOPE FOR FURTHER WORK

The concluding remarks along with the scope for further research are made in Chapter 7.

Afer Chapter 7, two appendices are included. Additional results from Chapter 2 are presented in Appendix A, and computational aspects of measures provided in Chapter 6 are discussed in Appendix B. These appendices would provide a better clarity of the results presented in the main text of the thesis.

Chapter 2

Stemming for Text Preprocessing of Web Documents

2.1 Introduction

Given a web page, one would find contents in the form of plain text, as well as, hypertext markup like tags for formatting the text, hyperlinks leading to other pages, or tags for embedding multimedia content. This chapter is concerned with preprocessing the textual content of web pages. The need for preprocessing textual content arises as it may be noisy due to various reasons, some of them being:

- Words spelt wrongly or in an alternate way
- Presence of stopwords or irrelevant words
- Use of several related but different words
- Presence of unexpected or foreign words
- Improperly formed or unclosed tags

Each one needs to be treated in its own way as they would otherwise interfere in basic tasks such as tokenizing, indexing, and retrieval. Misspelt words and word variants may

be detected using a pre-defined dictionary, and corrected using either a table lookup, or choosing valid words with a small edit distance from these words [80]. Stopwords may be removed by using an exclusion list [126], though in some studies, they are retained because of the value they provide in terms of context [89]. Similarly, some studies (e.g., [125]) ignore HTML tags for the sake of simplicity, while others (e.g., [87]) retain them because of the richness they provide to the textual contents. Taking care of related words is probably the most difficult and challenging part of preprocessing text data. Part of the challenge lies in defining what "related" means, and the other part is to find groups of words which can be clubbed together. We shall study in detail this task, known as stemming, in the remainder of this chapter.

2.2 Stemming

Stemming is the process of clubbing together words that are similar in nature. Generally, morphologically similar words are grouped together under the assumption that they are also semantically similar. Stemming is frequently used in the field of information retrieval [77, 128], because it results in an increase in recall, as documents that do not contain the exact query terms are also retrieved. In particular, all documents containing words with the same stem as the query term are considered relevant. Stemming also reduces the size of the feature set (when words are viewed as the features of documents). For the purpose of classification, this means that the models involved are far less complex than what would have been if the original set of words were used. This also means that it would lead to better generalization, in the sense that a small training error would imply a small test error too. It has been observed that the classification performance does not go down much due to the application of some of the standard stemmers. Also, this would lead to a reduction in the size of the index that needs to be stored. One may note that stemming in text mining may also be viewed as feature or prototype selection/reduction or clustering in pattern recognition, and as case selection in case based reasoning problems, where the

basic objective is to select the most representative features or dimensions or cases of a class or a concept based on some similarity measure or grouping.

Several standard techniques are available in the literature which perform stemming [50]. The strength of a stemmer is the amount of reduction in the size of the dictionary obtained by it [50]. Strong (or aggressive) stemmers may reduce the size of the index for a given corpus drastically. However, stemming is afflicted with two kinds of errors: under-stemming and over-stemming [147]. Under-stemming is the case where words that should have been grouped into the same class are not so, and the performance is suboptimal. When too many unrelated words are merged together, then it is the case of over-stemming. This leads to a reduction in precision during retrieval and an increase in the error rate for classification. One may also note that with the increase in the strength of a stemmer, recall is increased but either retrieval precision or classification accuracy (or both) are degraded. This may be concluded by observing that increase in stemming strength gradually leads to reduced number of stem classes, and in the extreme case all the words belong to a single stem class, which provides none of the information required for classification or retrieval.

Thus, the general objective in designing a stemmer is to ensure that the classification accuracy, as well as, the retrieval precision are maintained. In this article, we describe the design of such a stemmer. We make use of the classification information of the corpus, and model words as arising from a multinomial distribution [69]. A segregation method based on this can be employed on any existing rule based stemmer to refine its equivalence classes for improvement. The proposed methodology is found to improve both the classification accuracy and retrieval precision when applied on the Porter [118] and Truncate(3) [144] stemmers, and, compared with some existing ones, including the co-occurrence based refinement [144], and a distributional clustering based stemmer [6]. The classification performance is measured on the 20 Newsgroups and WebKB data sets, both in terms of accuracy and precision-recall plots of Naive Bayes, Support Vector Machines and Maximum Entropy based classifiers. The retrieval efficiency has been tested

on the Wall Street Journal (WSJ) data set, and precision-recall values have been displayed graphically. All the results have been tested for statistically significant improvement by the proposed methodology over some of the related methods.

The article is organized as follows. The background on stemming and related work is provided in Section 2.3. Then, we describe the proposed stemming technique in Sections 2.4 and 2.5 and present the experimental results in Section 2.6, respectively. We draw our conclusions in Section 2.7.

2.3 Stemming and Related Work

Documents are generally represented in terms of the words they contain, as in the vector space model [127]. Many of these words are similar to each other in the sense that they denote the same concept(s), i.e., they are semantically similar. Generally, morphologically similar words have similar semantic interpretations, though there are several exceptions to this, and may be considered as equivalent. The construction of such equivalence classes is known as stemming. A number of stemming algorithms, or stemmers, have been developed, which attempt to reduce a word to its stem or root form. Thus, the document may now be represented by the stems rather than by the original words. As the variants of a term are now conflated to a single representative form, it also reduces the dictionary size, which is the number of distinct terms needed for representing a set of documents. A smaller dictionary size results in a saving of storage space and processing time.

Stemming is often used in information retrieval because of the various advantages it provides [77]. The literature is divided on this aspect with some authors finding stemming helpful for retrieval tasks [77] while others did not find any advantage [59]. However, they are all unanimous regarding other advantages of stemming. Not only are the storage space for the corpus and retrieval times reduced, recall is increased without much loss of precision. Moreover, the system has the option of query expansion to help a user refine his/her query.

2.3.1 Different Stemming Algorithms

Various stemmers are available for several languages, including English. The most prominent ones are those introduced by Lovins, Dawson, Porter, Krovetz, Paice/Husk and Xu and Croft. We now provide a brief description of some of these algorithms.

Truncate(n)

This is a trivial stemmer that stems any word to the first n letters. It is also referred to as n-gram stemmer [144]. This is a very strong stemmer. However, when n is small, say, one or two, the number of over-stemming errors is huge. For this reason, it is mainly of academic interest only. In the present work, we have considered the value of $n = 3$, which makes the stemming very aggressive and refer to it as Trunc3.

Lovins Stemmer

The Lovins Stemmer [95] was developed by J. B. Lovins and is a single pass, longest match stemmer. It performs a lookup from a table of 294 endings, which have been arranged on a longest match principle. The Lovins Stemmer removes the longest suffix from a word. Once the ending is removed, the word is recoded using a different table which makes various adjustments to convert these stems into valid words. However, it is highly unreliable and frequently fails to form words from the stems, or match the stems of like meaning words.

Dawson Stemmer

The Dawson stemmer [38], developed by J.L. Dawson, extends the Lovins stemmer. It uses a much more comprehensive list of around 1,200 suffixes, organized as a set of branched character trees for rapid access. This, too, is a single pass and longest match algorithm. In this case there is no recoding stage, which had been found to be unreliable.

Porter Stemmer

Martin Porter proposed the Porter stemmer [118] which is based on the idea that the suffixes in the English language (approximately 1200) are mostly made up of a combination of smaller and simpler suffixes. It has five steps and within each step rules are applied until one of them passes the conditions. If a rule is accepted, the suffix is removed accordingly and the next step is performed. The resultant stem at the end of the fifth step is returned.

The Porter stemmer is very widely used and various implementations are available online at `http://www.tartarus.org/~martin/PorterStemmer/`. Versions of this stemmer are also available for non-English languages.

Paice/Husk Stemmer

The Paice/Husk stemmer [110] is a simple iterative stemmer, and uses just one table of rules; each rule may specify either deletion or replacement of an ending. The rules are grouped into sections corresponding to the final letter of the suffix making the access to the rule table quicker. Within each section the ordering of the rules is significant. Some rules are restricted to words from which no ending has yet been removed. After a rule has been applied, processing may be allowed to continue iteratively, or may be terminated.

Krovetz Stemmer

The Krovetz stemmer [78] was developed by R. Krovetz and makes use of inflectional linguistic morphology. It effectively and accurately removes inflectional suffixes in three steps, the conversion of a plural to its singular form, the conversion of past to present tense, and the removal of '-ing'. The conversion process firstly removes the suffix, and then through a process of checking in a dictionary for any recoding, returns the stem to a word. It is a light stemmer in comparison to the Porter and Paice/Husk stemmers.

Co-occurrence based stemmer by Xu and Croft

Xu and Croft [144] observed that most stemmers perform under-stemming or over-stemming, or even both. Strong stemmers generally perform over-stemming only. Xu and Croft came up with an algorithm that would refine the stemming performed by a strong stemmer. To this end, they computed the co-occurrences of pairs of words belonging to the same equivalence class. For each pair, they also computed the expected number of co-occurrences, which would account for the words occurring together randomly. Thus, they obtained a measure similar to the mutual information measure defined as:

$$em(w_i, w_j) = \max\left(\frac{n(i,j) - En(i,j)}{n_i + n_j}, 0\right),$$

where, n_i and n_j are the frequencies of w_i and w_j and $n(i,j)$ is the number of times the two words co-occur. E denotes the expected value. This measure ignores any co-occurrences that may be attributed to pure chance. Only if $em(w_i, w_j)$ is significantly greater than zero, they conclude that, in the given corpus, the two words indeed appear together and may be retained in the same equivalence class.

Splitting the equivalence classes in an optimal way however is computationally very expensive. When the equivalence classes are large, Xu and Croft opt for a suboptimal solution obtained by a connected component labeling algorithm applied after thresholding the em scores.

Dictionary based stemmers

There also have been dictionary based stemmers [54,71,77] which improve on an existing stemmer by employing knowledge obtained from a dictionary. Word co-occurrences in a dictionary are considered to imply the relations between words.

Probabilistic stemmers

Given a word in a corpus, the most likely suffix-prefix pair that constitutes the word is computed [5]. Each word is assumed to be made up of a stem (suffix) and a derivation

(prefix), and the joint probability of the (stem, derivation) pair is maximized over all possible pairs constituting the word. The suffix and prefix are chosen to be non-empty substrings of the given word, and it is not clear what should be done in the case when a word should be stemmed to itself.

Refinement of an existing stemmer

In some cases, errors produced by a stemmer are manually rectified by providing an exception list [78]. The stemmer would first look up the exception list and, if the word is found there, returns the stem found there. Otherwise, it uses the usual stemmer. The co-occurrence based stemmer mentioned above is also one such algorithm, where the exceptions are obtained automatically.

Distributional clustering as stemming

Distributional clustering [6] joins similar words into a group if the words have similar probability distributions among the target features that co-occur with them. In their work on document classification, Baker and McCallum had chosen the class labels as the target features. The root forms of the words are not taken into consideration while grouping them. This algorithm described in [6] is as follows. The mutual information of each word in the corpus with the class variable is computed, and the words are sorted in descending order. The number of desired clusters is fixed beforehand, say to M. The first M words are initialized to form M singleton clusters. The two most similar (of the M) clusters are merged. This similarity is measured in terms of the Kullback-Leibler divergence of the distributions of the two clusters. The next word in the sorted list forms a new singleton cluster. Thus, the number of clusters remains M each time. In the present work, we refer to this method as *baker*. In our implementation, we have fixed M to the number of stems obtained by refining the Truncate(3) stemmer using our model.

2.3.2 Stemming and Classification

There are several works on text classification (see, for example, [148]), where stemming has been employed in a routine manner. However, there are differences in opinions of researchers regarding the effectiveness of stemming for the purpose of classification. While Riloff [124] and Spitters [131] had concluded that stemming may not help increase classification accuracy, Buseman had observed that morphological analysis increases the performance for a series of classification algorithms applied to German email classification. In a recent work, Gaustad and Bouma [54] observed that stemming does not consistently improve classification accuracy. More recently, however, Cohen, et al. [30] found stemming advantageous while classifying medical documents.

Perhaps, the reasons for such varied observations lie in the different characteristics of the document collections involved. On the one hand, stemming would increase the number of instances per feature (by reducing the number of features), which is a favorable situation for classification. On the other hand, stemming may merge words regardless of the class information that they hold, thereby confusing a classifier which is presented with such mixed instances.

In what follows, we propose a novel stemming technique, whereby, even very strong stemming does not reduce the classification accuracy.

2.4 Proposed Stemming Technique

2.4.1 Criteria

We try to improve upon the existing stemmers discussed above on the following aspects:

1. Substitute words: these words, though very similar in meaning and/or usage, often do not tend to appear with each other. These are often the result of varying author styles, where a particular author uses just one of the substitute words all the time. Examples include words that have different spellings under British and American

usage (e.g., colour and color). To infer that they should be stemmed to the same word, one would need to analyze their co-occurrence with other words to find such relations.

2. Words with many senses: when a word has many senses, there might be words that are semantically similar to it in just one sense. Merging them would lead to loss of information [78]. It is desirable that only those words which match in all the given senses are merged.

3. Creation of new words: rule based stemmers occasionally create new words while stripping suffixes. For example, the Porter stemmer stems both *change* and *changing* to *chang*. Note that this is inevitable, and if a rule were to modify the final stem by adding an *e* to it, it would lead to yet other problems like *hang* and *hanging* both stemming to *hange*. The creation of such words may also increase ambiguity when dissimilar classes of words are merged. For example, the Porter stemmer stems *range*, *ranged*, *ranges*, *ranging* and *rang* to *rang*, even though *rang* is unrelated to the rest.

4. Simplicity and speed: rule based stemmers only need to step through a sequence of predefined rules and are very efficient, albeit at the cost of stemming errors. Corpus-based refinements are computationally expensive, as seen in the case of co-occurrence based stemmers, where the process of refining the stems involves computing the co-occurrences of each pair of words that map to the same stem. If an equivalence class (set of words mapping to the same stem) is "large", splitting it optimally becomes an arduous task.

5. Cross-corpus stemming: it is desirable to perform the stemming operation only once. Also, additional information like categories may not be available for all corpora. However, one would like the stemmer to perform reasonably well in that situation too. The stemmer may be built based on a single corpus and the same set

of stems is employed for other corpora, too. The challenge is to come up with a stemmer which does well even when the two corpora are very different in nature.

We now describe a stemming algorithm that incorporates all the above desiderata.

2.4.2 Stemmer refinement by distribution based segregation

The objective at hand is that given an equivalence class of words, it is to be split in such a way that the resulting equivalence classes reflect an improved stemming in terms of classification and retrieval. The primary objective is not to group morphological or semantically similar words, as a human linguist would do, though, such a feature would be added attraction. We utilize the information available in a classified text corpus to perform the splitting. The primary assumption behind the proposed methodology is that two words may be stemmed to the same stem if they are extremely similar in their distribution across various categories.

Each word is assumed to have a multinomial distribution [69] over the set of categories of the given corpus. In a multinomial distribution, n events are observed, each of which has k possible outcomes, with the i^{th} outcome having a probability of p_i. The binomial distribution is a special case where $k = 2$. Words deemed to be arising from the same multinomial distribution are kept in the same equivalence class, whereas, those which are significantly different from each other are separated out. Here, differences in the total number of appearances of the words (denoted by n) are ignored, just as in the vector space (or the bag of words) model. The distribution of each word is estimated from its frequencies in the various categories. Formally, the proposed methodology is as described below.

Let $\{w_1, w_2, \ldots, w_n\}$, be the set of words belonging to an equivalence class, i.e., they all stem to the same stem. Let K be the number of categories of the given text corpus. For each word w_i, we compute the occurrence vector $n_{i1}, n_{i2}, \ldots, n_{iK}$, where n_{ik} is the number of occurrences of the w_i under the kth category. We assume that each w_i

arises from a multinomial distribution whose parameters are $p_{i1}, p_{i2}, \ldots, p_{iK}$, and $n_i = \sum_{k=1}^{K} n_{ik}$. Here, each p_{ik} denotes the probability of w_i appearing under the kth category and is estimated as the corresponding proportion of occurrences in the corpus, n_{ik}/n_i.

The aim is to partition this set of words into non-empty subsets such that each subset consists of words whose estimated distributions are not significantly different from each other. Moreover, this task needs to be done without a prior knowledge of the size of the partition.

We employ a procedure similar to sequential hypothesis testing [137] for attaining this goal. Two thresholds/cutoffs, say t_1 and t_2, ($t_1 <= t_2$), are chosen for this purpose. The words are sorted in descending order of their frequencies. Without loss of generality, we shall now denote this sorted list of words by $\{w_1, w_2, \ldots, w_n\}$. The most frequent word, w_1, is chosen and is considered to stem to itself. We denote this as $stem(w_1) = s_1$. Let S be the current set of stems. So, initially, $S = \{s_1\}$. We shall also denote the equivalence class of stem s_j by S_j, defined as $S_j = \{w_k : stem(w_k) = s_j\}$.

For each subsequent word, we compute a distance between its distribution function and that of each stem in S, $d_{ij} = d(w_i, s_j)$. The distance function may be any of those discussed in Section 2.4.3. If each of these distances is greater than the bigger cutoff, i.e., $d_{ij} > t_2 \; \forall j$, we shall call the current word as a new stem and add it to the set S. On the other hand, if any of the distances, say d_{ij}, is smaller than the smaller cutoff t_1, we shall add the current word to the equivalence class of s_j, so that $stem(w_i) = s_j$.

This procedure is iterated with the two thresholds modified such that the new lower threshold is greater than t_1 and the larger one is smaller than t_2. It may be noted that, since the proposed algorithm depends on the accurate estimation of word distributions, the larger the number of words or documents per class, the better the expected performance.

For the purpose of cross-corpus stemming, the stems are first constructed based on one corpus. Then, the words of the other corpus are stemmed using the proposed method whenever they are available in the first one. For all other words, we fall back upon a standard stemmer like Porter or Trunc3.

2.4.3 Choice of distance function and thresholds

The above mentioned description provides a general form of the corpus based stemmer. For implementation, one needs to have a proper choice of distance function and thresholds. These are described below. The term 'distance' above can be defined in various ways to produce a variety of (mostly similar) stemmers. For example, the distance between a candidate word w_i and a stem s_j may be defined in one of the following ways:

- the distance between w_i and a prototype (or a representative) of the set S_j.
- the minimum distance between w_i and an element of S_j.
- the maximum distance between w_i and an element of S_j.

For each of the above options, we can also have one or a combination of the following kinds of distance functions:

- Euclidean distance between the distributions of the two
- Cosine distance (derived from the Cosine Similarity metric) of the distributions of the two [126]
- Kullback-Leibler distance between the distributions of the two [81]
- A test statistic that would be used for testing the equality of the two distributions [85]

The distance function may also take into consideration the size of the longest common prefix, so that words with a longer common prefix would be more likely to be stemmed to the same stem.

We deduce the computation time of our algorithm by looking at the operations performed for refining each of the initial equivalence classes. Suppose a stem class consists of n words and m concept groups. So, the objective is to split the given stem class into m concept classes. The words are sorted in descending order of their frequencies in

$O(n \log n)$ time. Then each word would be compared with at most m prototypes (one for each concept class). Thus, splitting a stem group is an $O(mn)$ operation (assuming $m < \log n$, otherwise, it would be $O(n \log n)$). It may be noted that, at this stage, co-occurrence based refinement would need to compute co-occurrences between all pairs of words, thereby becoming an $O(n^2)$ algorithm.

Now, if there are M stem classes initially, the complexity of our method is $O(Mmn)$. Also, n is expected to be $\frac{N}{M}$, where N is the total number of words, i.e., $Mn = N$. Hence, the average complexity of the proposed method is $O(mN)$, with m being interpreted as the average number of concept groups per initial stem class. We note that, in the above derivation, m depends on M because as M decreases the size of the initial equivalence classes, and consequently m, are expected to be increase.

This is an added advantage for the proposed method over the co-occurrence based method as it would not require a prior stemming result to start with for the refinement process (equivalent to using the Truncate(0) stemmer). However, for the purpose of stemming, this would necessitate the incorporation of the longest common prefix based modification during distance calculations.

There are no strict guidelines for choosing t_1 and t_2 except that a high t_1 would result in more words getting a stem class of their own (understemming), and a low t_2 would lead to large equivalence classes (overstemming). So, all that we are doing by choosing t_1 and t_2 is to fix a level of permissible understemming and overstemming errors. However, it is not possible to directly compute the exact number/proportion of such errors (since the exact distributions of the words are not known beforehand). If $t_1 = t_2$, then we do not need multiple iterations in the given procedure. This would result in a reduction in computing time. However, it may miss out on some simple mergers of equivalence classes. This is so because, once a word is called a new stem, it cannot be merged with any of the existing stems at a later stage. Choosing $t_1 < t_2$ allows us to do just that. In this case, whenever one is sure of neither merging the current word with an existing stem nor assigning it to a new class of its own, this decision may be put off for later. In

a following iteration, due to the change in the structure of the classes or the values of the chosen thresholds, the decision may become clearer. The strength of the stemmer would be proportional to the size of the thresholds t_1 and t_2.

2.5 Implementation

For the implementation of the proposed methodology, we preprocess the given corpus first and then refine a given stemmer by splitting the equivalence classes generated by that stemmer. We briefly describe these tasks here.

Any text corpus would contain several noisy terms. To clean such noise, some standard preprocessing tasks are performed on the given corpora. The headers of the documents are ignored altogether and only those words are retained which appear in at least two documents. HTML tags and stopwords are removed before building the model. All words are converted to lower case.

Then, to decide if a given word w_i may be merged with a stem class S_j, we test the difference between the estimated distributions of w_i and S_j. The distributions are estimated as the corresponding frequencies of words appearing under the K topics of the given corpus. For testing the difference between the two distributions, Pearson's statistic [85] is computed as described below. Let, $(n_{i1}, n_{i2}, \ldots, n_{iK})$ be the topic vector of w_i. Define m_{jk} to be the sum $\sum_{w_i \in S_j} n_{ik}$. Also, let m_j denote the total $\sum_{k=1}^{K} m_{jk}$. It is assumed that the estimated distribution of S_j is the actual one. To test if $(n_{i1}, n_{i2}, \ldots, n_{iK})$ has arisen from the distribution of S_j, Pearson statistic is computed as:

$$\frac{m_j}{n_i} \sum_{k=1}^{K} \frac{n_{ik}^2}{m_{jk}} - n_i$$

where, n_i and m_j are the totals, as defined above. When n_i is large, this statistic is known to approximately follow a χ^2 distribution with $K-1$ degrees of freedom.

Since some of the n_{ik} values may be zero, we replace them by

$$n'_{ik} = 0.9 n_{ik} + 0.1 \frac{n_i}{K} = n_{ik} + 0.1 \left(\frac{n_i}{K} - n_{ik} \right). \tag{2.1}$$

This is done for each term in the dictionary. What we do here is, essentially, to perform a smoothing operation. Now, none of the cells are empty and, moreover, the total remains the same. In the remainder of this article, we shall refer to n'_{ik} as n_{ik} itself. If w_i is merged with S_j, the m_{jk}'s are updated by adding n_{ik} (after modifying as in Eq. 2.1) for each j.

Since we have sorted the words in descending order of their frequencies, the $\chi^2_{(K-1)}$ assumption is satisfied initially. And, when the frequency of a word is very low, it would not matter too much as the word itself might not have much of a say during classification.

Splitting each equivalence class is performed in two iterations. The values of t_1 and t_2 are set to $\chi^2_{(K-1),\alpha}$ and $4\chi^2_{(K-1),\alpha}$, respectively, during the first iteration. Here, $\chi^2_{(K-1),\alpha}$ is the upper α cut-off of the $\chi^2_{(K-1)}$ distribution, i.e., the value of the $\chi^2_{(K-1)}$ distribution function at the chosen cut-off is $1-\alpha$. t_1 and t_2 are both set to $2\chi^2_{(K-1),\alpha}$ during the second iteration. In our implementation, we had chosen α to be 0.05.

Though our methodology does not create any new words of its own, during cross-corpus stemming, when the words encountered are not in the dictionary, a standard stemmer is used, and that may introduce new words into the system.

2.6 Experimental Results and Comparison

2.6.1 Data Sets Used

We evaluated the performance of the proposed methodology on two data sets, namely, 20 Newsgroups and WebKB. They are described below.

20 Newsgroups [1]

The 20 newsgroups (also known as 20NG) collection is a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. The 20NG data set is a collection of 19,997 newsgroup documents, partitioned evenly across 20 different newsgroups. The categories are listed in Table 2.1.

Table 2.1: List of Categories in the 20NG Data Set

alt.atheism	rec.sport.hockey
comp.graphics	sci.crypt
comp.os.ms-windows.misc	sci.electronics
comp.sys.ibm.pc.hardware	sci.med
comp.sys.mac.hardware	sci.space
comp.windows.x	soc.religion.christian
misc.forsale	talk.politics.guns
rec.autos	talk.politics.mideast
rec.motorcycles	talk.politics.misc
rec.sport.baseball	talk.religion.misc

WebKB [138]

This data set consists of web pages collected from computer science departments of various universities in January 1997 by the World Wide Web Knowledge Base project of the CMU text learning group. There are 8,282 pages and they were manually classified into the following seven categories (the figures in parentheses denote the number of pages in a particular category):

- student (1641)
- faculty (1124)
- staff (137)
- department (182)
- course (930)
- project (504)

- other (3764)

The class other is a collection of pages that were not deemed the “main page” representing an instance of the previous six classes.

For each class the data set contains pages from the four universities

- Cornell (867)
- Texas (827)
- Washington (1205)
- Wisconsin (1263)

and 4,120 miscellaneous pages collected from other universities.

WSJ

The Wall Street Journal data set is a part of the TREC collection [58], and consists of more than 170,000 records which appeared during 1987 to 1992 in the Wall Street Journal. The queries (also called topics) and query relevance scores (in the form of qrels files) are available at http://trec.nist.gov/data/test_coll.html.

2.6.2 Evaluation Procedure

The performance of distribution based stemmer refinement has been evaluated in two ways. First, a direct evaluation in terms of linguistic analysis has been performed. This would reveal how similar the system is to a human who groups together morphologically and semantically related words. The second evaluation is an indirect evaluation that observes the effects of stemming on classification accuracy and retrieval performance.

For performing the linguistic analysis, we followed the procedure described in [111]. A generalization of this procedure is provided in [39] and is useful for automatic evaluation of stemmers, but this is not employed here due to lack of resources on the authors’

part. There are 13,621 words in the intersection of the vocabularies of the webKB and 20NG data sets and a unix word list (generally, located at /usr/share/dict/words). Of these words, we chose all the words starting with the alphabets a, b, c, p, q and r, which comprised a total of 5235 words. These words were manually grouped into 2069 classes, on the basis that all and only those words which were judged to be semantically and morphologically related were kept in the same group. As in [111], words with at least the first two letters in common were considered for grouping together. So, words like *ran* and *run* or *buy* and *bought* were not stemmed to each other, but *bring* and *brought* were kept in the same group.

Paice has defined the following indices for quantifying overstemming and understemming. Let W be the size of the given word sample, and let N_G and N_S denote the number of concept groups (denoted by g) and stem classes (denoted by s), respectively. Also, let n_g and n_s denote the number of words in g and s, respectively. Now, suppose that g consists of words from k_g distinct stem classes, with u_{gi} instances from i^{th} such class, and that s consists of words from l_s distinct concept groups, with v_{si} instances from j^{th} such group, the understemming index (UI) and the overstemming index (OI) are defined as follows.

$$UI = \frac{\frac{1}{2}\sum_{g=1}^{N_G}\sum_{i=1}^{k_g} u_i(n_g - u_i)}{\frac{1}{2}\sum_{g=1}^{N_G} n_g(n_g - 1)} = \frac{\sum_{g=1}^{N_G} n_g^2 - \sum_{g=1}^{N_G}\sum_{i=1}^{k_g} u_{gi}^2}{\sum_{g=1}^{N_G} n_g^2 - W} \tag{2.2}$$

$$OI = \frac{\frac{1}{2}\sum_{s=1}^{N_S}\sum_{j=1}^{l_s} v_j(n_s - v_j)}{\frac{1}{2}\sum_{g=1}^{N_G} n_g(W - n_g)} = \frac{\sum_{s=1}^{N_S} n_s^2 - \sum_{s=1}^{N_S}\sum_{j=1}^{l_s} v_{sj}^2}{W^2 - \sum_{g=1}^{N_G} n_g^2} \tag{2.3}$$

UI and OI values were computed for all the stemmers based on both the 20NG and the webKB data sets. These values are displayed graphically in Figs. 2.1 and 2.2. A dashed line is drawn from the UI-OI values of Truncate(3) to those of $trunc3\_d$ its refinement to show how the errors changed on refinement. It may appear from the figures that while refining the Truncate(3) stemmer, too many understemming errors are introduced at the cost of reducing a few overstemming errors. However, this is not the case. It looks so just because the denominator in Eq. 2.3 is much larger than that in Eq. 2.2. More insight may

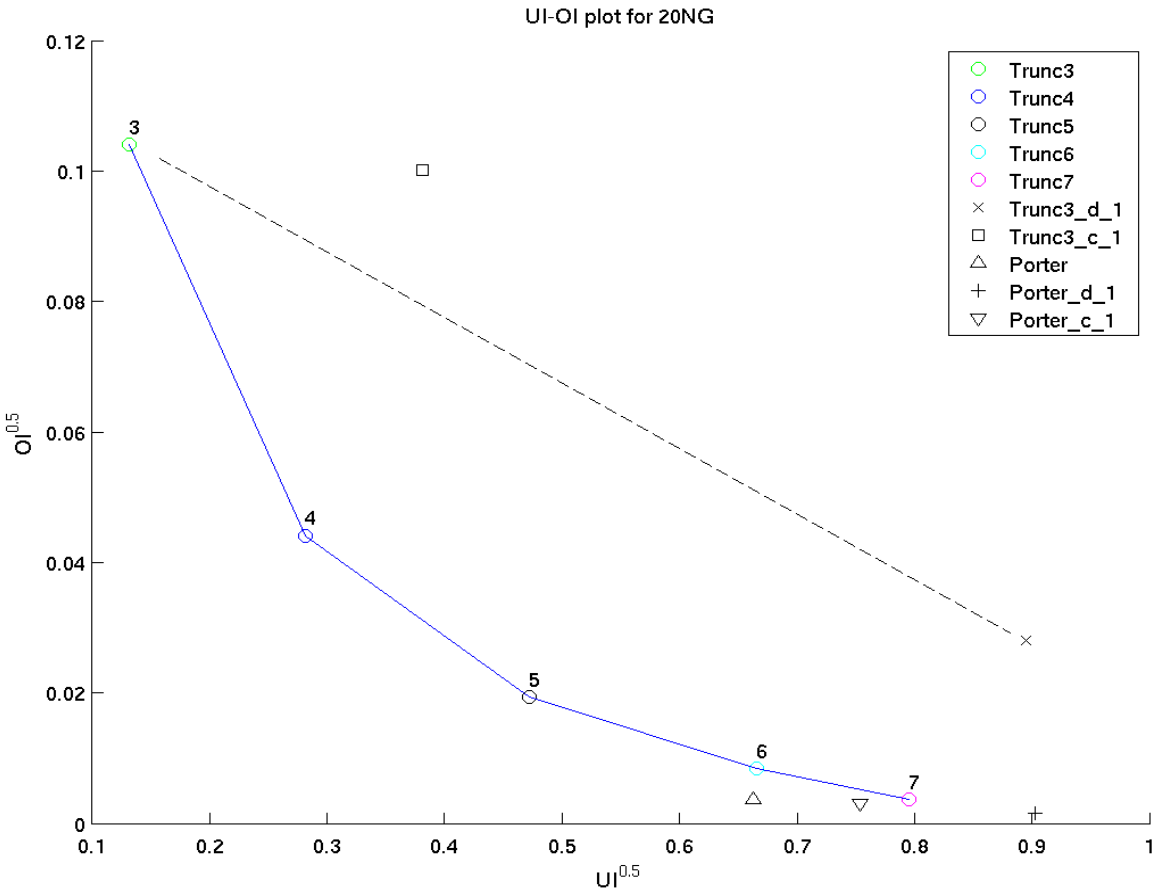


Figure 2.1: Plot of OI vs. UI for stemmers refined using 20NG data set: UI and OI values raised to the power $\frac{1}{2}$ for clarity

be gathered by looking at the *ang* example provided below.

We provide some examples of the equivalence classes produced by the proposed methodology by refining the Porter and Truncate(3) stemmers. The Porter stem class containing (*abort, aborts, aborted, abortion*) was split into two classes, whereby *abortion* was separated from the rest. Similarly, (*circularity*) was segregated from (*circular, circulars*). The stem group corresponding to *close* was split into three groups (*close, closing, closes*), (*closed*) and (*closely, closeness*). The above splits resulted from the differences in usages of the words in the collection. For example, even though semantically

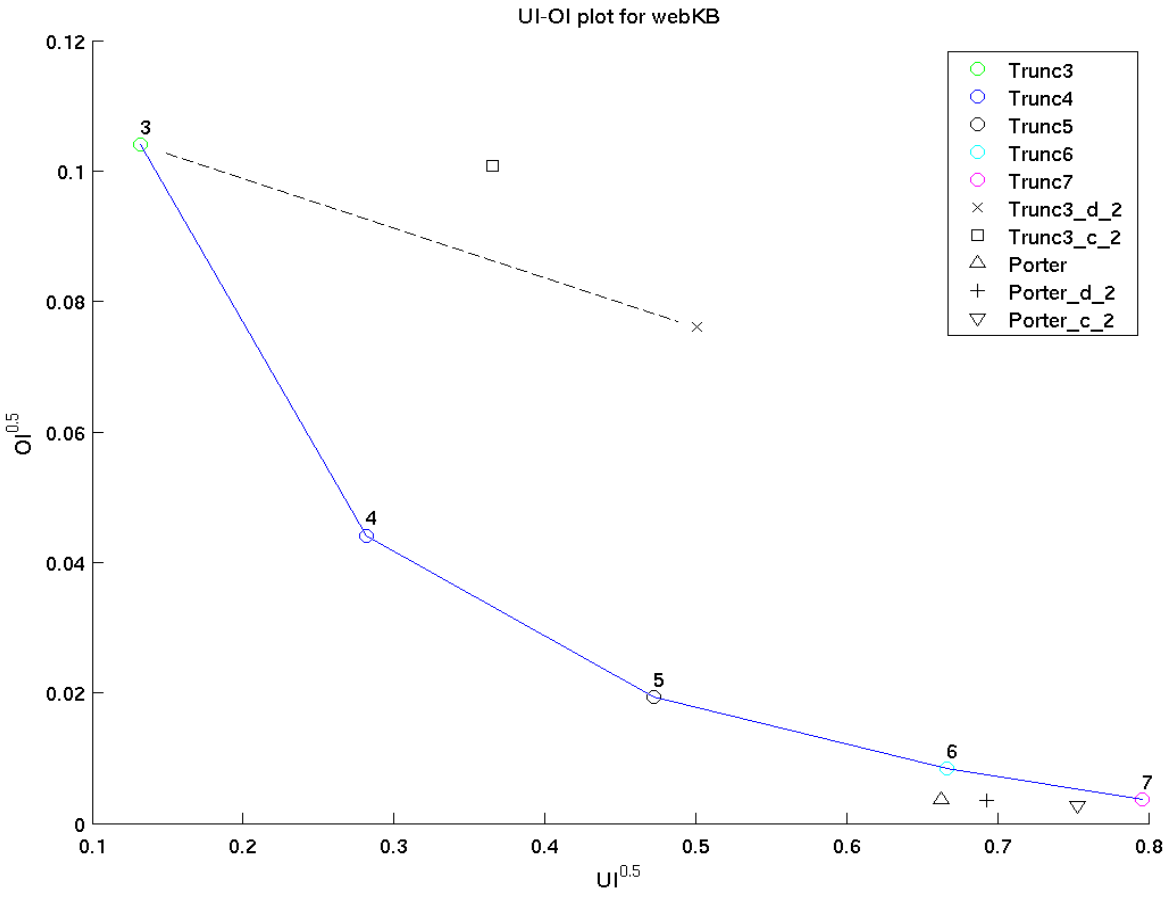


Figure 2.2: Plot of OI vs. UI for stemmers refined using webKB data set: UI and OI values raised to the power $\frac{1}{2}$ for clarity

and morphologically, *circularity* is related to *circular*, this term (and its plural) have a different meaning in general and arise in a different context. Similarly, though the verb and adjective forms of *close* have been separated out, *closed* was made into a new group of itself, resulting in an understemming error.

Next, we present an example of refining the Truncate(3) stemmer. The stem class *ang* is split into the following eight classes (the first word of each group is the stem). *(angel, angelic, anglican, anglo, angling, anguish, anglicans), (angeles, angelo), (angelino), (angels), (anger, angry, angola, angered, angelos, ang, angrier, angering), (angers), (angle, angles, angular, angst, angus, angled, angulated, angstrom), (angmar).* As may be seen, despite some overstemming (e.g., *angling* mixed with *angel*) and understemming (e.g., *angels* and *angers* are left out as singletons instead of being merged with *angel* and *anger*, respectively) errors, the splitting is largely successful as most of the related words appear in the same group, especially because no linguistic analysis is performed. In particular, it is interesting to note that *angstrom* has been retained in the same group as *angle*, perhaps, as a consequence of both appearing in similar contexts.

The detailed analysis of the overstemming and understemming errors after refining the stem class *ang* is presented here. We treat this set of 29 words as the whole sample available to us. There are 8 stem classes, and 14 concept groups consisting of these words. 8 of the 29 words are proper nouns. The calculations for GDMT, GUMT, GDNT and GWMT are provided in Tables 2.2 and 2.3. Hence, we see that, $UI = \frac{13}{37} = 0.35$ and $\frac{54}{369} = 0.15$.

For evaluating the performance of our system in refining the classification accuracy of a stemmer we used the Bow Toolkit [96] and conducted the following experiments for each data set. The document collections are preprocessed as mentioned in Section 2.5, and the equivalence classes are split accordingly.

For training and testing, the data set is split randomly into two parts and the same split is used for each stemmer. The proportion of documents chosen for training is first taken to be 60% and later the experiment was repeated with the same chosen to be 40%. The

Table 2.2: The 14 concept groups for words starting with *ang*

g	n_g	k_g	$(u_1, \ldots, u_{k_g})$	DMT_g	UMT_g	DNT_g
ang	1	1	1	0	0	14
angel, angelic, angels	3	2	2, 1	3	2	39
angeles	1	1	1	0	0	14
angelino	1	1	1	0	0	14
angelo, angelos	2	2	1, 1	1	1	27
anger, angered, angering, angers, angrier, angry	6	2	5, 1	15	5	69
angle, angled, angles, angling, angular, angulated	6	2	5, 1	15	5	69
anglican, anglicans, anglo	3	1	1	3	0	39
angmar	1	1	1	0	0	14
angola	1	1	1	0	0	14
angst	1	1	1	0	0	14
angstrom	1	1	1	0	0	14
anguish	1	1	1	0	0	14
angus	1	1	1	0	0	14
Total:	29			37	13	369

Table 2.3: The 8 stem classes for words starting with *ang*

s	n_s	l_s	$(v_1, \ldots, v_{l_s})$	WMT_s
angel, angelic — anglican, anglicans, anglo — angling — anguish	7	4	2, 3, 1, 1	17
angeles — angelo	2	2	1, 1	1
angelino	1	1	1	0
angels	1	1	1	0
anger, angered, angering, angrier, angry — ang — angelos — angola	8	4	5, 1, 1, 1	18
angle, angled, angles, angular, angulated — angst — angstrom — angus	8	4	5, 1, 1, 1	18
angmar	1	1	1	0
angus	1	1	1	0
Total:	29			54

training and testing phases are repeated five times for each choice of the proportion.

The text classification algorithms employed to compute the classification accuracy were Naive Bayes (NB) [88], Support Vector Machines (SVM) [68, 135], and Maximum Entropy Method (MaxEnt) [106]. Each document in the test set was given a classification score for each of the available categories. If a single category was to be assigned to a document, the one with the maximum score for that document was chosen. We computed the classification accuracy which is the proportion of test documents assigned to the correct class. This value was computed for each of the individual categories also.

Classification accuracy measures the total number of correctly classified documents. However, when documents are misclassified, it does not distinguish between them on the basis of their classification scores. To take this into account, we have adopted the following precision-recall method. The documents are first sorted in descending order of the classification scores. Only the largest score was considered for each document. Now, at any value of recall, the precision (or classification accuracy) is computed. A higher precision-recall curve is preferable. It may be noted that classification accuracy can be obtained from this curve as the precision when recall is set to 100%. Experiments are also performed where the precision-recall curves are obtained for each individual category.

To evaluate the retrieval capabilities of our algorithm, we conducted the following experiments on the Wall Street Journal data set using the SMART system. Topics 101 to 150 were chosen as the given queries. The word vector weighting was set to TFIDF. For each query, documents are retrieved and the precision is noted at recall values set to $10\%, 20\%, \ldots, 100\%$. These precision values are averaged over all queries and are presented in the form of precision-recall plots.

2.6.3 Comparison

As described in Section 2.4.2, the proposed distribution based segregation methodology can be employed to refine the equivalence classes generated by any existing stemmer. In the present investigation, we used it to obtain new stemmers based on the Porter and Trun-

cate(3) stemming for both the corpora. Let these new stemmers be denoted as *porter_d_*i and *trunc3_d_*i, where i is 1 for the 20NG data set and 2 for the WebKB data set. Similarly, the stemmers derived using Baker and McCallum's distributional clustering are denoted as *baker_1* and *baker_2*.

The reason for choosing Porter and Truncate(3) stemmers is as described below. Porter's stemmer is one of the most standard stemmers as is evident by its use in the literature. Truncate(3) is a stemmer that is used mostly for academic purposes. We have used that because it is a very strong stemmer and results in several over-stemming errors, thereby providing one a significant of scope for refinement.

The comparison process has four parts. In the first part, we compare the performance, in terms of the classification accuracies, of the refined new stemmers *porter_d_*i and *trunc3_d_*i with that of the original ones (i.e., *porter* and *trunc3*), as well as, no stemming and *baker_*i. The objective is to demonstrate both the effectiveness of refinement by our method, and improvement over the baseline performance where the original words are used.

In the second part, we compare these new stemmers with the co-occurrence based modified Porter and Trunc3 stemming of Xu and Croft [144]. These may be denoted, in short, as *porter_c_*i and *trunc3_c_*i, i being the same as in the first part. Here the objective is to compare the performance of our distribution based refinement process with the co-occurrence based refinement process. These stemmers are also compared with *baker_*i.

The third part deals with comparison in terms of cross-corpus performance, where a stemmer refined using the information from one data set is applied to another data set. The objective is to study the dependence of a stemming algorithm on the data set based on which it is derived, and its applicability to a dissimilar data. Here, we consider *trunc3_d_1* and *trunc3_c_1* applied to the WebKB data set, and *trunc3_d_2* and *trunc3_c_2* applied to the 20NG data set.

The fourth part involves comparison with respect to retrieval performance, where the objective is to study the effects of the stemmers on the retrieval at various levels of recall.

The stemmers being considered here are *porter_c_1*, *porter_d_1*, *trunc3_d_1* and *baker_1*, and they are applied to the WSJ data set. The stemmers refined on the 20NG data set are chosen for retrieval on the WSJ data set in order to evaluate how the refinements generalize to other data sets in the case of retrieval.

2.6.4 Results

Tables 2.5 and 2.6 report the classification accuracies obtained by different stemming algorithms for the 20NG data set and the WebKB data set, respectively. All abbreviations used in the result tables are described in Subsection 2.6.3. We make the following observations from Tables 2.5 and 2.6:

- both *porter_d* and *trunc3_d* fare better than *porter* and *trunc3*, respectively, in all cases for both the data sets.
- *porter_d* shows a better performance than *porter_c* in all the cases, though the number of stems obtained by *porter_d_1* is slightly more than that by *porter_c_1*.
- *trunc3_d* provides a better classification accuracy compared to *no stemming* and *baker*, the case of SVM for the WebKB data set being the only exception. The same observation holds when *trunc3_d* is compared to *porter_d*.
- the number of words common to both the data sets is 20782, which is about 64% and 37% of the dictionary sizes of the WebKB and 20NG data sets, respectively (see Table 2.4). The classification accuracy obtained by *trunc3_d_1* is significantly better than *trunc3_c_1* when applied to the WebKB data set. This confirms that the refinement procedure performed by employing the classification information from a different corpus works better when the number of common words is large.
- The statistical significance of the improvement gained by using the proposed stemmers over that of existing stemmers is tested using a t-statistic. Table 2.8 contains the $t-statistic$ values for the case of the Naive Bayes classifier applied to the 20NG

data set when the test set size is chosen to be 40% (Fig. 1). Table 2.9 shows the corresponding values for the WebKB data set. We observe from Table 2.8, which corresponds to Fig. 1, that there is a significant improvement, at 95% confidence level, over existing stemmers when *trunc3_d_1* is employed. Similarly, *trunc3_d_2* has significantly, at the 95% confidence level, outperformed existing stemmers as can be seen from Table 2.9, which corresponds to Fig. 5.

Apart from the classification accuracies, we also provide results in the form of precision-recall plots (Figs. 2.3 and 2.4). More plots, corresponding to different classification methods, are available but have been moved to Appendix A (Figs. A.1–A.6) to avoid cluttering the present chapter. It may be noted that the proposed algorithm consistently outperforms the rest, especially, when considered along with the number of stems obtained (Table 2.4). When based on *trunc3*, the proposed method (i.e., *trunc3_d*) consistently improves the classification precision for all values of recall. This effect is more prominent when Naive Bayes is used for classification (Figs. 2.3, A.1, 2.4 and A.4). This is possibly because the stems obtained by *trunc3_d* satisfy the Naive Bayes' independence assumption better than the original set of words. In that respect, our methodology may be considered as a "modification of feature sets to make the independence assumption more true" as mentioned in [88].

Note here that the time taken for stemming words is the same for any of the stemmers, as the words and their stems can be stored in a hash table. It is the time taken for creating this hash table that may differ. While this is not significant for any of the rule based stemmers, it is comparatively quite high for the co-occurrence based stemmer, the proposed one, as well as, *baker* (Table 2.7). The steep increase in the computation time for *trunc3_c* is a consequence of the large equivalence classes formed by *trunc3*.

For testing the retrieval efficiency of the proposed methodology, we have chosen a part of the WSJ data set consisting of Wall Street Journal articles published between 1987 to 1992. The queries used were topics 101 to 150. The retrieval experiments have been performed using the SMART system with the word vector weighting set to TFIDF [26].

Table 2.4: Stem counts and the largest equivalence classes obtained by various stemming algorithms

Stemming Method	Data Set					
	20NG			WebKB		
	Stem Count	Largest Class	Index Compression	Stem Count	Largest Class	Index Compression
no stemming	56436			32299		
porter	40821	gener: 24	13.6%	23446	gener: 25	15.3%
trunc3	8158	con: 675	76.6%	5053	con: 432	81.6%
trunc4	21252	inte: 236	42.2%	13283	inte: 192	50.4%
trunc5	33187	inter: 174	24.4%	20051	inter: 147	31.6%
porter_c	47441	generic: 18	11.0%	26249	general: 24	25.6%
trunc3_c	30710	considered: 652	24.2%	14921	convex: 430	44.8%
porter_d	48502	general: 13	10.7%	23971	general: 25	27.0%
trunc3_d	22643	discussion: 212	41.2%	6163	contents: 261	72.5%
trunc3_c (cross)	13771	convex: 360	47.3%	11036	considered: 358	56.9%
trunc3_d (cross)	10708	con: 314	52.2%	13857	int: 85	49.5%

Table 2.5: Classification Accuracies for 20NG

Stemming Method	Classification Method		
	NaiveBayes	SVM	MaxEnt
no stemming	80.44	80.06	77.08
porter	79.37	79.73	77.40
trunc3	72.88	71.11	68.24
trunc4	78.68	77.41	73.64
trunc5	79.73	78.15	74.24
porter_c_1	79.59	79.57	77.46
trunc3_c_1	74.28	74.13	75.03
porter_d_1	80.34	80.21	77.53
trunc3_d_1	81.98	81.19	77.85
baker_1	81.38	81.19	77.15
trunc3_c_2	74.82	73.38	71.01
trunc3_d_2	74.73	73.51	70.59

Table 2.6: Classification Accuracies for WebKB

Stemming Method	Classification Method		
	NaiveBayes	SVM	MaxEnt
no stemming	63.02	72.35	64.09
porter	61.86	70.14	62.76
trunc3	57.55	63.77	58.07
trunc4	60.85	66.56	50.67
trunc5	61.27	64.57	51.13
porter_c_2	62.00	70.16	62.64
trunc3_c_2	59.90	65.11	58.11
porter_d_2	63.08	71.38	63.91
trunc3_d_2	63.95	69.53	64.21
baker_2	61.19	69.51	64.43
trunc3_c_1	59.10	64.77	58.60
trunc3_d_1	61.39	69.51	64.13

Table 2.7: Time (in secs) taken for creating the stem hash tables

Method	Data Set	
	20NG	WebKB
porter_d	3.2	2.6
trunc3_d	5.5	5.1
porter_c	3.6	2.2
trunc3_c	9.2	4.7
baker	11.2	9.4

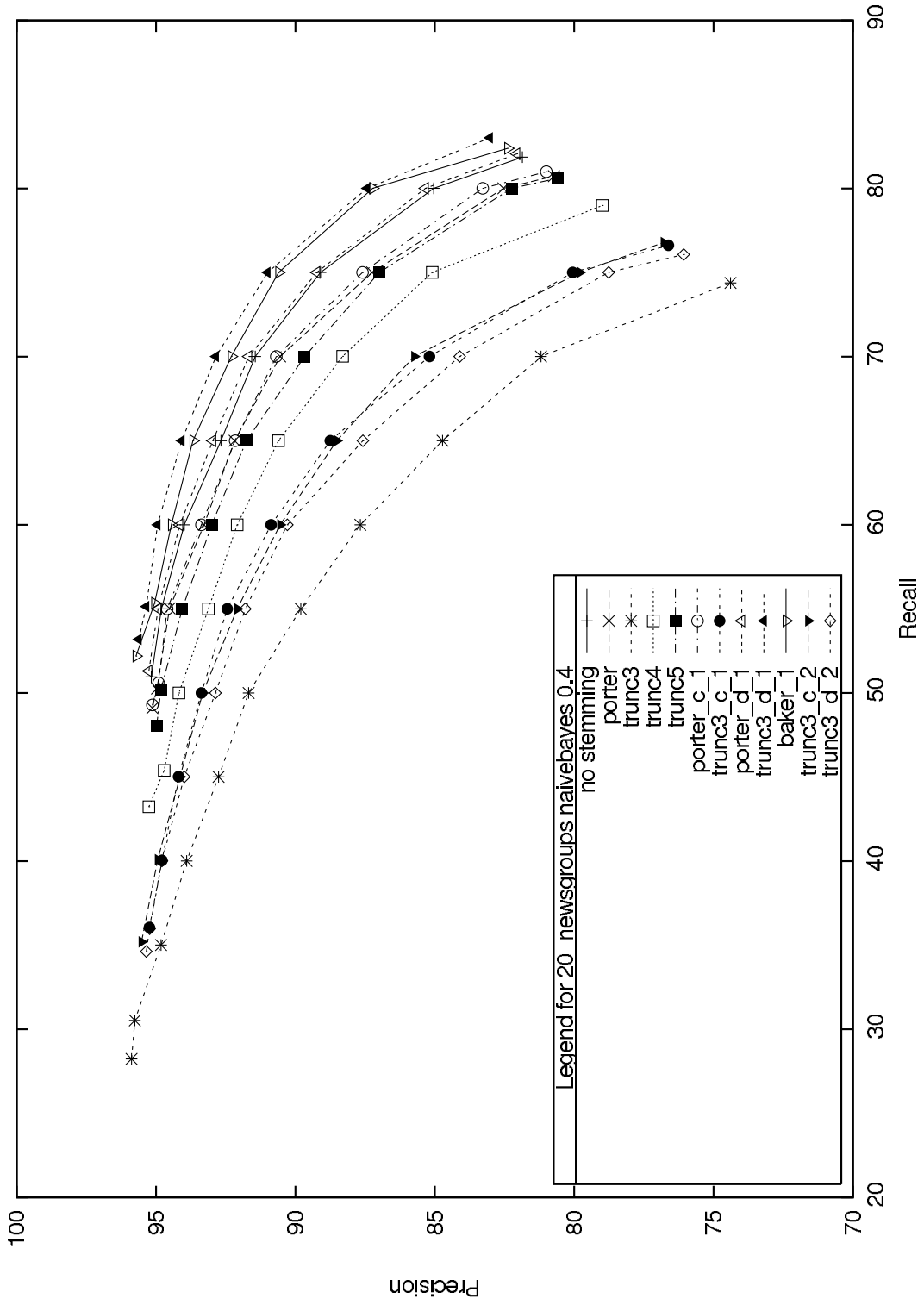


Figure 2.3: Data Set: 20NG, Test Set Size: 40%, Method: NB.

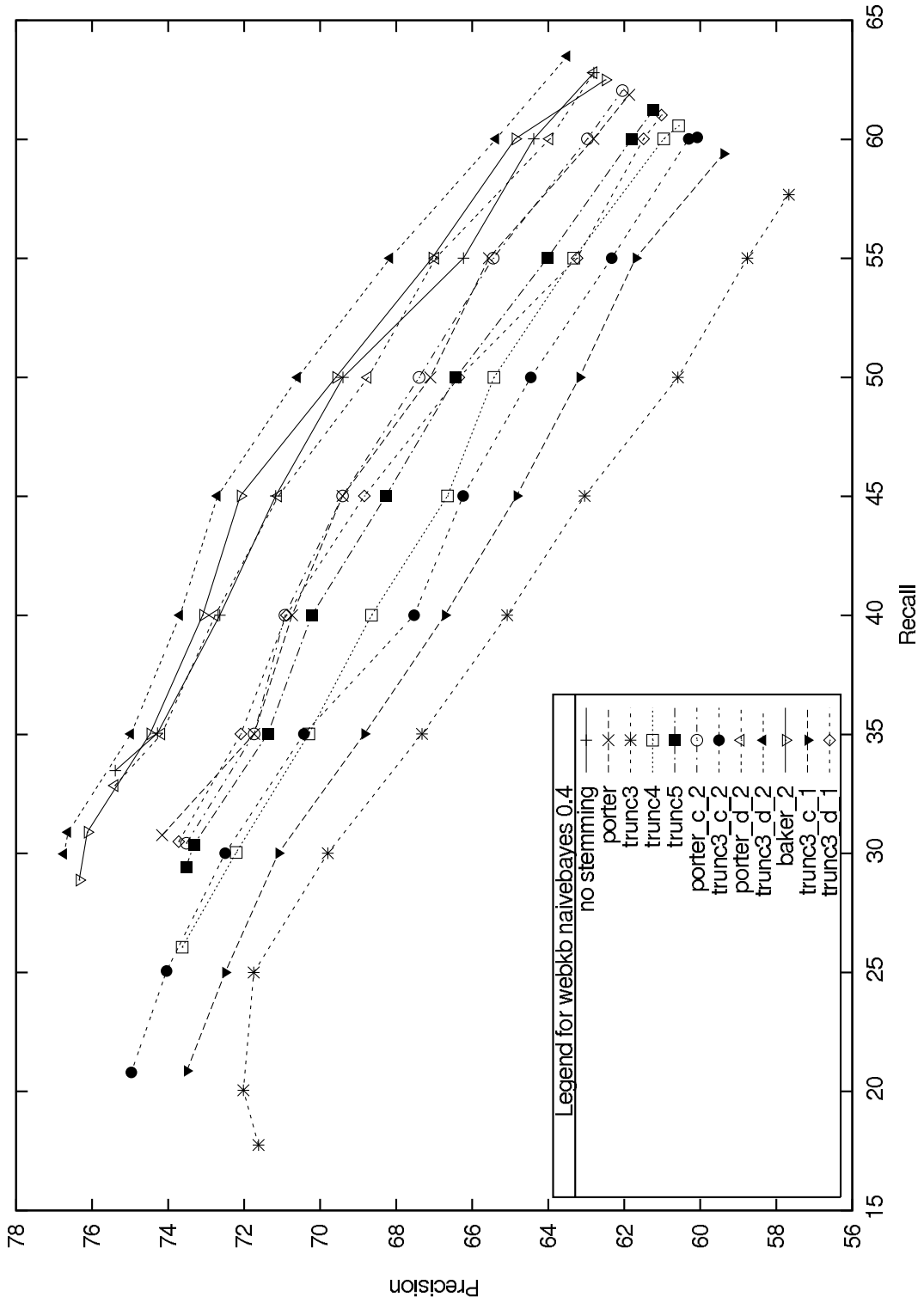


Figure 2.4: Data Set: WebKB, Test Set Size: 40%, Method: NB.

Table 2.8: Statistical Significance values for 20NG using NB

Existing Stemmers	Proposed Stemmers		
	porter_d_1	trunc3_d_1	trunc3_d_2
no stemming	0.17 (0.43)	2.45 (0.02)	0.27 (0.40)
porter	1.73 (0.06)	4.38 (0.00)	2.53 (0.02)
trunc3	11.43 (0.00)	13.70 (0.00)	10.40 (0.00)
trunc4	9.23 (0.00)	9.91 (0.00)	8.89 (0.00)
trunc5	7.17 (0.00)	7.83 (0.00)	7.43 (0.00)
porter_c_1	2.53 (0.02)	4.21 (0.00)	2.29 (0.02)
trunc3_c_1	4.91 (0.00)	7.87 (0.00)	6.51 (0.00)
baker_1	1.31 (0.11)	2.17 (0.03)	1.21 (0.13)
trunc3_c_2	7.72 (0.00)	9.16 (0.00)	6.28 (0.00)

Table 2.9: Statistical Significance values for WebKB using NB

Existing Stemmers	Proposed Stemmers		
	porter_d_2	trunc3_d_2	trunc3_d_1
no stemming	0.87 (0.20)	2.43 (0.02)	-0.36 (0.64)
porter	4.13 (0.00)	5.70 (0.00)	-0.15 (0.56)
trunc3	13.21 (0.00)	15.18 (0.00)	5.64 (0.00)
trunc4	9.27 (0.00)	12.91 (0.00)	5.14 (0.00)
trunc5	8.41 (0.00)	11.58 (0.00)	4.73 (0.00)
porter_c_2	3.93 (0.02)	5.81 (0.00)	-0.26 (0.60)
trunc3_c_2	7.80 (0.00)	9.37 (0.00)	2.11 (0.03)
baker_2	0.71 (0.25)	1.97 (0.04)	-0.61 (0.72)
trunc3_c_1	9.45 (0.00)	10.96 (0.00)	2.97 (0.01)

As mentioned earlier in Section 2.6.2, we have used the refinements obtained from the 20NG data set. We note that *trunc3_d_1* outperforms *baker_1*, as well as the remaining methods, as seen from Fig. 2.5, with *trunc3_d* providing more than 2% improvement over *baker_1* until the recall is above 90%. The improvement has been found to be statistically significant at a 95% confidence level.

2.7 Conclusions

We have described the design of a stemming algorithm which uses the classification information of a corpus to refine a given stemmer. The main advantage over other stemmers like co-occurrence based stemmers is its ability to drastically reduce the dictionary size while maintaining both the classification accuracy and retrieval precision. Experiments conducted on 20NG and WebKB data sets confirm the superiority of the proposed methodology for the task of text categorization when classifiers like Naive Bayes, Support Vector Machines and Maximum Entropy Method are used. This is also supported by precision-recall based evaluation. Another set of experiments performed on WSJ data set demonstrates the enhancement in retrieval precision when the refined stemmers are employed instead of existing stemmers. The performance of refinement done by employing the classification information from a different corpus increases as the number of common words increases.

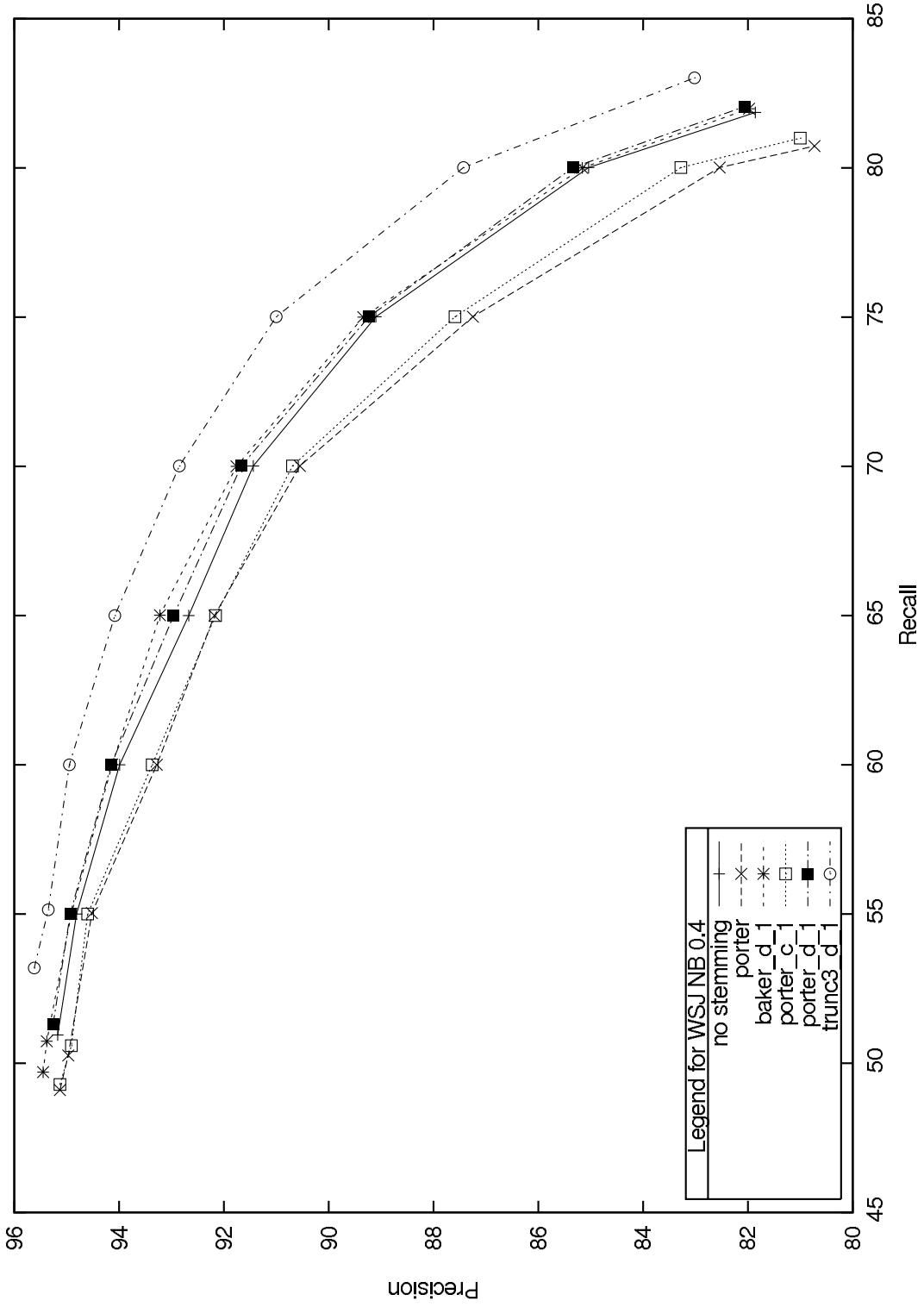


Figure 2.5: Retrieval results on WSJ, Similarity measure is TFIDF

Chapter 3

Sequence Detection for Link Preprocessing of Web Documents

3.1 Introduction

In the previous chapter, we had concerned ourselves with preprocessing the textual contents of individual web pages. Web pages, apart from containing text, also have a mechanism for connecting with other web pages through the use of hyperlinks. The present chapter deals with how web pages may be preprocessed based on the interconnections between them.

The Web has a complex graph structure and is usually modeled as a digraph [19, 22], with the pages forming the vertices and the links between them being the arcs. This graph structure, as reflected by the hyperlinks between web pages, carries a lot of information in addition to the contents of individual pages.

The author of a web page may want a visitor to move on to one of a small set of pages. On occasion, this set might be a singleton, implying that the visitor is urged to directly continue on to that page. Pages which are intended to be visited one after the other, in a particular order, shall hereafter be referred to as a *sequence*. A sequence is a path in the graph theoretic sense, but it is not just any path, but one where there is a continuity

in the contents of the constituent pages. A formal definition of *sequences*, along with the motivation for detecting them, is provided in Section 3.2.

A literature survey on sequence detection in graph theory, in Section 3.3, reveals that, while there are a plethora of algorithms for detecting cycles and paths (in the graph theoretic sense) in graphs, there is none that suits our specific purpose. We provide a novel algorithm that detects, not just any ordered set of distinct pages, but sequences of pages that were originally intended to be visited in that order. This algorithm involves detecting continuity links and finding pieces of the sequence that may be joined together later. We then make a few assumptions and produce a scalable but relaxed version of this algorithm. These algorithms are described in Sections 3.4 and 3.5.

The true worth of detecting sequences lies in understanding the numerous tasks that it helps in – document retrieval, web graph size reduction, and duplicate detection, to name a few. For all these tasks, the detected sequence may be replaced by a single web page that includes the contents of all the pages in the sequence. While the reduction of the size of the web graph is an obvious consequence, and duplicate detection is fairly easy to understand, the effects of merging documents on retrieval tasks are more complicated. We study thoroughly how the *term frequency* and *inverse document frequency* change as documents in the corpus are merged together. This constitutes Sections 3.6 and 3.7.

In Section 3.8, we provide results of experiments performed on HTML corpora that confirm the efficacy and usefulness of the work presented in the current chapter. Section 3.9 summarizes the contributions of the chapter and concludes it.

We now delve deeper into defining sequences of interest, and subsequently, identifying them from the vast web graph.

3.2 Sequences and Cycles of Web Pages

We use the following terminology related to graphs [57]. A digraph or directed graph G is an ordered pair (V, E), where V is a set of vertices and E is the set of arcs or

directed edges between these vertices. We denote the arc from vertex i to vertex j by e_{ij}. A path in a digraph is a sequence of vertices $x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_n$ such that x_i and x_{i+1} are connected by an arc in E, and the vertices are not repeated. A cycle is a sequence of vertices $x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_n, x_{n+1}$ such that $x_{n+1} = x_1$ and if x_{n+1} is removed, the remaining part is a path. We denote the path and cycle defined here by $x_1.x_2 \ldots x_i.x_{i+1} \ldots x_n$ and $(x_1.x_2 \ldots x_i.x_{i+1} \ldots x_n)$, respectively. A digraph is strongly connected if there exists a path from any vertex x_1 to any other vertex x_n in V. The in-degree (out-degree) of a vertex is the number of arcs leading to (going out of) the vertex.

The number of pages on the indexable Web is several billion, with Yahoo! claiming to have indexed about 20 billion pages way back in the year 2005 [146]. According to [2], the average number of links on a web page ranges between 9 and 28, which turns out to be a humongous number of links for the whole web (note that, some, or many, of these links may point to pages outside the indexed set of pages). Link analysis algorithms dealing with the web graph assume that it is strongly connected, although, in practice, this is ensured by adding artificial arcs to the graph [22].

In such massive digraphs, with several arcs between the vertices, sequences and cycles are a common phenomenon. We now present an example to illustrate how huge the number of cycles in the web graph could be, and why most of these individual cycles are not interesting.

3.2.1 Motivation

Consider the following statement.

A: There is a cycle of web documents consisting of

- `http://www.stanford.edu/index.html`,
- `http://www.yahoo.com/index.html`, and
- `http://wi-consortium.org/index.html`

One can easily be convinced that Statement A is true, primarily because of the high in- and out-degrees of the chosen web pages. Moreover, Statement A would hold true even in conjunction with any of the following statements:

B: The three pages appear in a prespecified order

C: The length of the cycle is exactly 10

D: The cycle contains no web page from a `co.uk` or a `.net` domain

The ease with which these statements have been made, and can be verified to be true, provides an idea of how huge the number of cycles involving these three web pages would be. This in turn would imply that the total number of cycles on the web (without imposing any conditions) would be extremely large. As mentioned earlier, the web has more than 20 billion documents, and an average out-degree of 20. Assuming this to be an Erdos-Renyi (random) digraph $G(n, p)$, with n being the total number of documents, and p being the proportion of all possible links which actually exist, the expected number of cycles of length, say 10, turns out to be of the order of 20^{10}. Enumerating all such cycles (say, by an algorithm as mentioned in [133] or [94]) would be tedious and computationally expensive, and several individual cycles that have been formed just by chance may not be interesting at all.

Despite the huge number of cycles on the web, only a few of these cycles were conceived at the time of their creation by the authors of the constituent pages. While the reason behind creating such special structures in the web graph may be Search Engine Optimization or simply a matter of convenience, sometimes it borders on malicious intent or spam [142]. Moreover, with web authoring styles varying widely, the same kind of content may be presented in a single page, or broken up into several pages. Such dissimilarities bring about large differences in link based analyses. So, for the sake of uniformity and consistency during comparison of web pages, the detection of these special structures is essential. We now specify what exactly are the structures of our interest.

3.2.2 Structures of Interest

We first take the help of an example (see Fig. 3.1), to characterize the kind of objects we are interested in detecting from the web graph. Fig. 3.1 shows a directed graph, say $\mathcal{G}$,

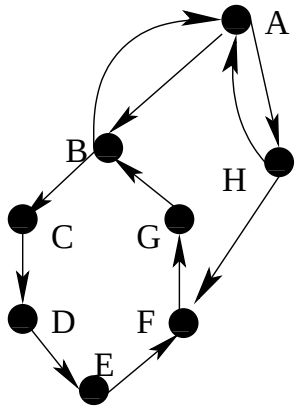


Figure 3.1: A strongly connected digraph

with the vertices labeled A to I being web documents and arcs being the links between them. This graph is strongly connected because any vertex can be reached from any other vertex.

We now look at the directed cycles of the graph $\mathcal{G}$. There are two of them of length greater than two, namely (B.C.D.E.F.G.B) and (A.I.H.F.G.B.A), each consisting of six vertices. Although, technically, both the above cycles may be reported by a cycle detection algorithm, we observe that the former looks more regular compared to the latter. Also, the names of the web pages are quite suggestive that the first cycle was intended right at the outset, while the second one was formed just by chance.

We are interested only in cycles of the first kind which exhibit a regular pattern. In the present chapter, we study sequences and cycles of web pages that have been deliberately so generated by their author(s). In other words, at the time of creation of these pages, it was intended that a surfer would view them all in a specific order.

We note that although our examples involved cycles of a graph, we could have made similar statements about paths of the same graph. As we shall see later, paths are what

offer the most interesting applications in our context and cycle detection provides no additional advantages for the tasks being considered in the present chapter.

We shall now formalize the definition of sequences of interest, which as we already know, are not just any paths in the given graph. Mathematically, a sequence $\{X_n\}$ is defined as a function with its domain being a subset of $\mathcal{N}$ (the set of non-negative integers) and taking values in some set S. In our case, each value is a web page. The kind of sequences that we are interested in may be defined as an ordered set of elements where the relation between any two consecutive elements remains the same. In other words, X_{n+1} is related to X_n in exactly the same way as X_n is related to X_{n-1}.

A *sequence* of web pages is an ordered set of web pages $x_1.x_2.\ldots.x_k$, such that consecutive pages of a sequence are connected by *continuity links*, and x_{i+2} is linked to x_{i+1} just the way x_{i+1} is to x_i. To understand continuity links better, let us look at the various relationships bestowed upon two pages connected by a hyperlink. Hyperlinks play a variety of roles like

- Reference: the user is expected to follow the hyperlink, visit the reference and return back to the present page. Such instances abound in sites like Wikipedia [141].

- Continuation: the link is to content which the author of the present page would recommend the user to access next. For example, a link from `http://www.mutt.org/doc/manual/manual-1.html` to `http://www.mutt.org/doc/manual/manual-2.html` is a continuity link.

- Navigation: links of this kind are present to help the user easily navigate between portions of the site. Usually, the same set of such navigational links (possibly, excluding a self-link) appears in each page of a section of a site, resulting in a clique of web pages.

- Advertisement: these links lead the visitor of the current page to advertisements, which may or may not result in a (financial) gain to the owner of the web page.

The past few years have seen increasingly contextually relevant advertisements being served, with the links being created dynamically at the time of page generation. However, several studies (e.g., [150]) treat a link to an advertisement as noise, assuming that it is forcibly imposed on the user.

The above mentioned categories are not necessarily mutually exclusive (and are not meant to be exhaustive, either). In particular, it is difficult to accurately determine if a link is for the purpose of continuation or navigation. For example, Bharat and Mihaila treat any link between two web pages in the same domain as navigational [17]. This definition would include, to various extents, all four types of links described above. In particular, when continuity links connect pages in the same domain (or perhaps, within the same directory of a server), they could be flagged as navigational links.

We now make two important assumptions regarding continuity links. Put otherwise, these are assumptions regarding how an author would split her content across multiple pages.

- *Assumption 1*: Links for continuity occur only between pages in the same domain. In other words, continuity of content occurs only through navigational links. This assumption is usually reasonable as an author is most likely to split content between pages on the same domain.

- *Assumption 2*: Content is continued on pages at the same directory level. This is an extension of the previous assumption. Sequential content would generally be split (or organized) into files under the same directory.

An example of continued content that satisfies these assumptions is `http://www.mutt.org/doc/manual/manual-[1-7].html`. These assumptions go a long way in reducing the computational complexity of sequence detection, because the search space for finding any sequence containing a particular page is restricted to the directory under which that page exists, as opposed to the whole web in the unrestricted case. In certain cases, content may be continued over several directories, and under

these restrictive assumptions, only parts of the content would be identified as being continued. For example, `http://www.mpt.iif.hu/pages/1/page1.html` and `http://www.mpt.iif.hu/pages/2/page1.html` are part of continued content. However, we would not treat them as part of a single sequence. In other words, we would not necessarily be finding maximal subsequences, which is a tradeoff for the gain in speed.

A consequence of Assumptions 1 and 2 is that continuity links are now a strict subset of navigational links which have been widely studied in the literature. In the next section, we review existing literature on detecting navigational links and graph theoretic cycles.

3.3 Related Work

Sequence detection algorithms described in this chapter rely on both identification of continuity links and detecting sequences and cycles from graphs. Here, we survey the literature on both these topics.

The concept of continuity links has not been studied elsewhere, and the closest related literature is regarding navigational links which, as mentioned earlier, contain the set of continuity links. Identification of navigational links has been studied in many works available in the literature [17, 20, 37, 149]. Bharat and Mihaila [17] disregarded links between pages on affiliated hosts. Two hosts were called affiliated if they shared the first three octets in their IP addresses, or the rightmost non-generic tokens in their hostnames was the same. Borodin, *et al.* [20] identified and eliminated navigational links using a very similar idea. However, not all navigational links are detected in this manner [20]. Moreover, this approach is quite severe, and some links connecting pages wholly on the basis of their content would be wrongly classified as navigational links. This would especially be the case where, say, a member of an organization links to content on a colleague's page.

Yi and Liu [150] detect navigational links along with banner ads, decoration pictures,

etc, which were collectively termed web page noise, in a set of pages by constructing a compressed structure tree (CST). The diversity of tags in an element node of the CST determines how noisy the corresponding block in the web page is.

In the above mentioned studies, navigational links have been treated as noise and discarded. Also, in [150], they have not been separated from other kinds of web page noise like banner ads, decoration pictures, *etc*.

Sequence or path detection in graphs has largely been restricted to finding Hamiltonian paths, shortest paths between a pair of vertices, or counting the total number of paths between two vertices. This conveys the impression that sequence detection, by itself, was not considered very interesting. On the other hand, cycle detection has been widely studied under two related areas, namely, pseudo-random number generation [21, 107] and graph theory [94, 140]. In pseudo-random number generation, a sequence of numbers is generated by applying the same function to the last generated number. The objective is to detect cycles in the sequences of random numbers being generated. By adding arcs between consecutive elements of such sequences, this problem may be studied as a graph cycle detection algorithm. Since, every element of such sequences has out-degree one, stack based algorithms are employed for cycle detection [107].

Detecting cycles in general graphs and digraphs is more complicated, because the out-degrees of the vertices may be more than one. There are several digraph cycle detection algorithms, as evident from the opening statement of [94], which was itself published several years back. Some of them like the one by Read and Tarzan depend on depth-first search while others like the Szwarcfiter-Lauer algorithm employ a recursive backtracking procedure which search strongly connected components of the graph [94]. For the web graph, a sizeable portion of which may form a single strongly connected component, such algorithms may only serve an academic purpose.

We now present algorithms that detect sequences of interest from the web graph.

3.4 Proposed Sequence Detection Technique

We now describe two methodologies for detecting sequences in a set of web documents.

- The first one relies directly on the hyperlink structure, and is very much in line with the definition of sequence mentioned above. In order to detect a sequence, its individual segments are detected first, whereby, each triplet (three consecutive elements) of the sequence is found. Such triplets are found by examining the similarity of the pairs of links e_{AB} and e_{BC}, where A, B and C are three web pages present in the web graph under consideration. Detection of the desired triplets of the form (A, B, C) is performed in two steps. First, for each page B, we identify the continuity links leading into and out of B. Then, each pair of links consisting of an inlink and an outlink are checked to see if they form a part of a sequence. All such triplets are stored and finally, they are merged to obtain sequences of web pages.

- The second one, which does not directly rely as much on the hyperlink structure, puts less emphasis on the accurate identification of continuity links, and is much more scalable. Sequences are determined by looking at just a single page, and the positions of the continuity links in that page. This is further simplified into looking at consecutive URLs found in the crawl order, and residing under the same directory. This assumes that the web crawling is performed in a breadth first manner. In that case, when consecutive continuity links appear in a "contents" page that has just been crawled, the corresponding pages would, usually, be crawled or traversed in that order.

We now describe the first method for detecting sequences of web pages. As a first step, we need to identify triplets of pages connected by continuity links. Since we do not know which links are continuity links, this is not directly possible. So, we shall identify candidate triplets connected by navigational links instead. However, given that our

ultimate goal is to select only those candidate triplets which are connected by continuity links, we shall consider only those navigational links which connect pages within the same directory.

Our approach for detecting navigational links is relatively simple compared to several other approaches mainly because we are not interested in various other structures like content blocks and templates, *etc*. We assume that navigational links mostly occur unaccompanied by text, the likely reason being that the anchor text is descriptive enough for the user to understand where the link leads to. Since most continuity links lead to neighboring pages, which in turn, have a similar look and feel, the user is generally expected to be familiar with the navigational links and no further description is required.

Whereas Bharat and Mihaila [17] treated links between any two pages on affiliated hosts as navigational, we shall consider navigational links only if both the source and target pages are at the same level, i.e., the two pages are located in the same directory under the same domain. In this manner, we restrict the sequences to be detected to consist of pages at the same level. This additional constraint in the definition of continuity links drastically reduces the complexity of the algorithm for searching sequences in the web graph.

For a given web page, we look at the link elements (delimited by " ... "), and content elements which may either be a text token or an image. We ignore all other content types in this study. A typical such sequence of text and links looks like in Fig. 3.2 (this particular example sequence corresponds to the URL `http://clgiles.ist.psu.edu/index.html`). Here, 'L' and 'T' denote a link and content element, respectively.

Note that the navigational links are clumped together, whereas, the other links in the page are surrounded by text. We formalize this notion as follows: Let the string formed as above be denoted by S and let its length be n. Also, let k be such that $S[k]$ is 'L'. Let the set $\{S[k-5], S[k-4], S[k-3], S[k-2], S[k-1]\}$ be called the left neighborhood of $S[k]$ and $\{S[k+1], S[k+2], S[k+3], S[k+4], S[k+5]\}$ be the right neighborhood

TTTTLLLLLLTLLTTLLTTTTTTTTTTTTTTTTTTTL
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTLTTLTTTT
TTTTLTTTTTTTTLTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTLTTTTTTTT

Figure 3.2: String of links and content elements of a web page

(one of them may have less than five elements, too, if $k < 6$ or $k > n - 6$). Now, if either the left or the right neighborhood of $S[k]$ contains at most three 'T's, then we label $S[k]$ as a navigational link. Otherwise, the link is considered to be a referential link and is ignored. Basically, we are looking at the two neighboring windows (of size 5) to see if it is predominantly text in that locality or not. The optimal choice of window size is not obvious, but we find that our present choice is reasonably good for our experiments.

Among the navigational links found, we further classify them into two kinds: top and bottom. We assume that a continuity link would appear at the ends of a page (and not at the middle). If k mentioned above is smaller (greater) than $\frac{n}{2}$, we label the link as top (bottom). This is our third simplistic assumption about continuity links:

- *Assumption 3*: Continuity links appear either at the top or the bottom of a page, and not within running text.

This assumption is inspired from the standard practice of including a link to "next" and "previous" pages at the top and/or bottom of each page. Note that by "top" we mean the top portion in the HTML version of the page, and not necessarily the top portion of the page when displayed in a browser. For example, the links in the initial portion of the HTML content of a page may appear as a bar to the left.

We now represent the features of the link by using a byte of information. The first (second) bit denotes whether the link is a top (bottom) continuity link. The first (latter) three of the remaining six bits keep the count of the link from the top (bottom). So, for example, the second navigational link from the top would have 1*001*** and the third navigational link from the bottom would have *1***010 (the values at '*' are determined by the position of the link at the other end of the page). We provide another example to clarify this. A navigational link that appears as the fourth link from the top, and appears again as the last link in a page, would have the information byte 11011000 associated with it. Since we use only three bits to store the count of the link, only the top and bottom eight navigational links are retained, and the rest are not called navigational links and the corresponding bits at the beginning are set to 0.

Now that the navigational links have been identified, our task is to find the successor, if any, of each such link. In the case, where we manage to find a successor, we shall call the navigational link as a continuity link. For each page B, we look at the navigational links that lead into B. For any such link e_{AB}, we look at its information byte and identify the link on B with the same information byte. If there exists one such link, say e_{BC}, we call it the successor of e_{AB} and the triplet is noted as $A.B.C$, and indexed by B for ease of retrieval. What we are doing in terms of finding the successor of a link is trying to identify the link that appears in almost the same position as e_{AB}. Since, the user has reached page B by clicking a link at that very position on A, it is very likely that the link on page B at the same position would be followed. This is what we mean by saying that page C is related to B in the same way as B is related to A.

In this manner, all links which are present in sequences are assigned a successor and are stored as triplets. The relation between the consecutive elements of the sequence might not hold at the extreme ends of the sequences, but all triplets from the interior would be found. These form the basic building blocks of larger sequences. Our task is now to concatenate these segments into the actual sequence. Let $x_1.x_2 \ldots x_n$ be an existing subsequence (initially, all of them are of length 3). A subsequence $y_1.y_2 \ldots y_m$ is prepended

to it if $y_{m-1} = x_1$ and $y_m = x_2$, and the new subsequence becomes $y_1 \ldots y_m . x_3 \ldots x_n$. Similarly, it is appended if $y_1 = x_{n-1}$ and $y_2 = x_n$ and then the new subsequence becomes $x_1 . x_2 \ldots x_n . y_3 \ldots y_m$.

We traverse these triplets, merging them into larger and large subsequences to obtain all the desired sequences. Finally, when we have all the sequences, we check if there are any cycles to be found. Note that this scheme of cycle detection corresponds to the one used for detecting cycles in sequences of pseudo-random numbers. We do not need the more general graph cycle finding algorithms because we are now dealing with only a sequence of web pages and not the whole web graph. We refer to this algorithm as SC1.

Continuity links may sometimes be marked out in the HTML content of the page. This is all the more common nowadays with a number of pages being generated automatically or being converted from other formats such as MSWord or LaTeX. If we assume that the appearance of the words *navigation* or *navig* in HTML comments or names of blocks genuinely indicates that the block is a navigation panel, identifying continuity links becomes straightforward. This obviates the tokenization of the pages and counting the neighbors of links, and hence, improves the speed of the algorithm. Some web pages may also provide information about the relations with other pages by means of `link` elements. This again makes the task of detecting sequences of pages trivial as the relations between them are specified beforehand. However, authenticity of these specified relations is not guaranteed, and an algorithm may be easily misled onto a wrong path.

The information about a continuity link that we store may not convey the exact position where it would be displayed due to a variety of reasons. A more accurate way would be to locate the actual position in the page where the link would be placed by analyzing the HTML structure. This information may require more than a byte of storage space. It may be noted that these storage requirements are far below that needed for the actual index itself.

The next section describes the second of the above mentioned methods for sequence detection.

3.5 Trading Accuracy for Scalability

Often the ideal definition of a sequence, which tries to gauge the positions of the links, might not be scalable as a wide variety of documents enter the corpus, and especially, if there are several documents under the same directory. The sequence detection algorithm would be sent on a wild goose chase more and more frequently.

This section aims at efficiently detecting a majority of the sequences, the occasional omission being traded in favor of scalability. Of course, these improvements in efficiency would be garnered an additional assumption on the kinds of sequences we shall try to find.

- *Assumption 4*: Names of the files (the last part of the URLs) are named so as to reflect the continuity in content.

Usually, this is achieved by using a numerical or alphabetical suffix before the file extension, with the suffix increasing one at a time.

This assumption enables us to identify sequences without looking inside the HTML contents of the indexed web pages. Only the list of URLs is needed, which may then be sorted. Note that sorting the URLs would automatically group together those that share a common directory. We also note that a simple unix sort would not always suffice for sorting when there are numeric suffixes. For example, a string comparison of page9.html with page10.html would place the former after the latter. We make a simple change to the string comparison algorithm to take care of these cases. If both file names have only numerals from the first position (from the left) where the two strings differ to the end of the strings, then the result of the string comparison would be the same as the comparison of these numeric suffixes. Otherwise, usual string comparison is performed.

Next, we run through all the sorted URLs, and whenever two consecutive URLs have the same directory name, we compute the distance between the two filenames. As earlier, if the differing portions are both numeric, we compute the arithmetic difference as the distance. Otherwise, we use the standard string distance function, namely, the Levenshtein

edit distance. If this distance is less than or equal to 2, we shall call the two URLs to be part of a sequence, and tag the second one with the same tag as the first one. Each URL in the list would thus be assigned a number corresponding to which sequence it belongs to, with some of these sequences being of length one. Since the first sequence detection algorithm, SC1, does not detect sequences of length 2, in the present case also, we forcibly ensure that sequences are of length at least three. From a practical point of view, and to be on the safer side, we also impose an upper bound of 25 on the length of permissible sequences. We call this algorithm SC2.

3.6 Characteristics and Uses

We now discuss some characteristics of the proposed sequence detection technique. SC1 and SC2 detect sequences, but do not report paths that have been formed just by chance. For example, these algorithms would surely not report a cycle that satisfies Statement A (of Section 3.2.1). Thus, despite the existence of numerous cycles in the web graph, these algorithms would report only a handful of sequences and cycles.

This methodology does not incorporate finding second level sequences and cycles, i.e., sequences of sequences and cycles, *etc*. This may be required for the sake of uniformity in the case where a page is split into not just a single sequence but into multiple sequences at different levels.

Note that we have taken care of only decimal suffixes in the SC2 algorithm. If the numeric prefixes are non-decimal, SC2 might not be able to detect some of the sequences. For example, if two consecutive filenames are part12.html and part20.html, where the numeric suffixes are ternary numbers, SC2 would not recognize them as part of a single sequence. However, there are instances where consecutive elements with non-decimal suffixes may be detected. For example, page9.html and pagea.html, where the 'a' has been used as the hexadecimal equivalent of 10, by virtue of having an edit distance of 1, these are tagged as being part of the same sequence. Such instances are usually found

when authors try to maintain a constant length for the file names. The condition on the minimum length of sequences rules out cases where content is split into exactly two parts (say, part1.html and part2.html). While, technically, such cases may be easily detected by relaxing this condition, we retain the minimum length condition because there is too little information to infer about the "next in sequence" property.

We now come to the utility of detecting sequences and cycles of web pages. One apparent use of detecting sequences is to merge the pages in that order. What were individual pages prior to merging, will now become sections of a single page. In this manner, the number of vertices in the web graph may be reduced, which is computationally advantageous.

The proposed methodology may also be employed for bringing a consistency into the computation of page ranks despite the varying authoring styles. Some web page authors may prefer to split their content into a sequence of pages, while others may like to keep it all in a single page. These differences have an impact on the page ranks of the pages. To see this, we revisit our example in Fig. 3.1, where, our cycle detection algorithm would detect only the cycle (B.C.D.E.F.G). Now, consider a page BB that has the union of the contents of all these individual six pages. We look at how the above mentioned cycle of pages would fare against the page BB with respect to page ranking. To this end, we create a new example (Fig. 3.3), made of two strongly connected components and connect them. We note that the two components of this directed graph, one consisting of the vertices AA, BB, HH and II, and the other with the remaining vertices, have a correspondence between them, with BB corresponding to the cycle (B.C.D.E.F.G). This graph is treated as the web graph and the (unnormalized) PageRanks [22] of these nodes are shown below (Table 3.1). Although, content-wise, the pages B to G, put together, are equivalent to the page BB, their ranks are different. In general, for the case of the actual web graph, it would be much more difficult to ascertain the effect of merging a sequence of pages, but, as evident from this example, it is not necessary that the ranks of the constituent pages remain the same. Thus, in the case of web search, the results returned may be influenced by whether

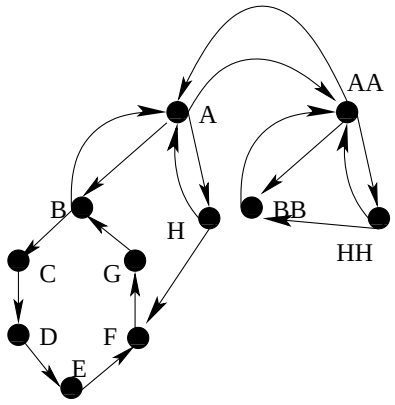


Figure 3.3: One component of the graph has a cycle, the other has it merged

all the content has been kept in a single page or has been split into a number of pages. In this respect, our algorithm may be utilized to maintain uniformity among various portions of the web, so that comparisons between web pages become less dependent on authoring styles.

There is yet another and more important advantage of merging such sequences of web pages. Consider two pages X and Y which form part of a sequence. Suppose that the pages X and Y contain terms t_1 and t_2, respectively, and are the most authoritative, individually, for the respective terms. Let Z be a page containing both t_1 and t_2, but which is not as authoritative on either t_1 or t_2 as X and Y, respectively. Z may be returned ahead of both X and Y as the most authoritative page containing both t_1 and t_2. Note that X may not even contain the term t_2, and even if it does, may not be authoritative for it. However, when both X and Y form part of a merged sequence, they would correctly be returned ahead of Z for the query consisting of t_1 and t_2. In reality, pages like Z behave like pages containing spam words. They are not at all authoritative for the terms they contain, but nevertheless have terms which co-occur very rarely. Such pages permeate the initial portion of the search results when a user enters such a combination of terms, primarily because of the lack of better pages containing all the query terms together.

It may be noted that the pages X and Y need not be physically merged together. A search engine may perform the analysis as if their content is combined together and output

Table 3.1: PageRanks for the pages in Fig 3.3

Page	Rank
A	1.172
B	1.542
C	0.805
D	0.834
E	0.859
F	1.290
G	1.247
H	0.482
AA	1.295
BB	0.956
HH	0.517

both the pages in response to a query. In this manner, sets of web pages may be output by a search engine in response to a (multiple term) query, as opposed to the current trend of providing single pages as search results.

Merging documents has a major impact on the retrieval process. We dig deeper into how the frequencies of the terms change as a result of merging documents.

3.7 Impact of Merging Documents on TFIDF Scores

A TFIDF based score is a normalized measure of the importance of a term t to a document d in the corpus. TF refers to Term Frequency, and is the count of the occurrences of the term in the document. Since, longer documents are likely to have more instances of the term t, TF is normalized in order to be able to compare across documents. A common

method of normalizing TF is to divide by the total number of tokens in the document [26].

$$tf(t,d) = \frac{n(t,d)}{\sum_\tau n(\tau,d)} \tag{3.1}$$

That some words occur more frequently in the corpus than others has to be taken into consideration while combining the importance scores of various words for a document. This is done by weighting the TF scores by the Inverse of the Document Frequency (IDF). IDF is inversely proportional to the number of documents in the corpus containing term t, and is generally defined as [26]:

$$idf(t) = log(\frac{1+|D|}{|D_t|}) \tag{3.2}$$

The TFIDF score of t for document d is simply taken as the product of the corresponding TF and IDF values.

Now, suppose that the given corpus is C and we have detected some sequences of documents and merged them together resulting in the corpus C'. We study how the TFIDF scores for the various terms and documents are changed as a consequence of merging the documents. Here, t is a term in a document d which is contained in the original corpus C, and d' is a document in C'. Let N and N' denote the number of documents in C and C', respectively. By definition, $N' \leq N$.

- If $d \in C'$ (that is, document d is unchanged as it is not part of any detected sequence), the TF of t in d is unchanged. However, the IDF of t may change as $\frac{DF'(t)}{N'}$ need not equal $\frac{DF(t)}{N}$. This is because

$$IDF(t) \uparrow \Leftrightarrow DF(t) \downarrow \Leftrightarrow \frac{DF(t)}{N} > \frac{DF'(t)}{N'} \Leftrightarrow \frac{N'}{N} > \frac{DF'(t)}{DF(t)}.$$

  In words, the IDF of t increases if a higher proportion of documents containing t are merged together, as compared to the proportion of documents merged in the original corpus. So, the overall importance of t for the document d, as reflected by $tfidf(t,d)$, increases in such a case, as a consequence of the term t now being expected to appear in fewer documents as compared to other terms in d. Similar

statements hold for the decrease, and no change, in importance of the term t for the document d.

- If $d \notin C'$ (that is, document d has been merged together with some other documents to form d'), the term frequency of a term t that occurs frequently in many of the documents merged together to form d' is higher than that of a term which appears rarely in all the documents put together. For example, if t_1 and t_2 both have equal term frequency in d, but t_1 appears in no other documents constituting d', whereas t_2 appears in all of those documents, the term frequency of t_1 in d' would be lower than that of t_2 in d'. Also, it is clear that t_2 would benefit more than t_1 in terms of increase in IDF (with or without normalization). Thus $tfidf(t, d')$ would be more for terms appearing more in the whole sequence of documents, which seems reasonable, as it means that the term that is contained in several of the documents is deemed to have higher importance for the collection as a whole.

- The previous points were concerned with comparison between terms. Now, we shall study the effect of merging different documents on a single term. If a term t appears once in each of the documents d_1, d_2 and d_3, and say, d_2 and d_3 are merged together, whereas d_1 is kept separate, then the (normalized) TF of t is still very similar to the unmerged situation. To elaborate, let us assume that n_1, n_2, and n_3 are the number of words in the three documents respectively. Now, the TF of t in d_1 remains $\frac{1}{n_1}$, whereas the TF of t in d_2 merged with d_3 is $\frac{2}{n_2+n_3}$. It may be noted that the quantity $\frac{2}{n_2+n_3}$ is the inverse of the arithmetic mean ($A.M.$) of n_2 and n_3, and $\frac{\frac{1}{n_2}+\frac{1}{n_3}}{2}$ is the inverse of the harmonic mean ($H.M.$) of n_2 and n_3, and hence, the following set of inequalities hold:

$$\min\left\{\frac{1}{n_2}, \frac{1}{n_3}\right\} \leq \frac{2}{n_2+n_3} \leq \frac{\frac{1}{n_2}+\frac{1}{n_3}}{2},$$

where the second inequality follows from $A.M. \geq H.M.$, and the first inequality

may be seen as follows:

$$\frac{n_2 + n_3}{2} \leq \max\{n_2, n_3\} = \frac{1}{\min\left\{\frac{1}{n_2}, \frac{1}{n_3}\right\}}.$$

Thus, the TF of t may not increase or decrease much when the documents containing t are merged together. However, even if the absolute TF of t is unchanged, as noted earlier, the TF of t would be relatively higher than the TF of some other terms. This is a consequence of there being more distinct terms in d_2 and d_3 put together, than individually.

3.8 Experimental Results

We now present the results of experiments. We have tested the proposed algorithm on three different data sets. First we test our algorithms on the Python data set, a small collection of web pages, so that a manual verification of the results is feasible. All files under the website `http://docs.python.org` were obtained (available online as a tar bzipped file from `http://python.fyxm.net/ftp/python/doc/2.4/html-2.4.tar.bz2`). There were a total of 1412 HTML files. Of these, 1408 HTML files were under 10 subdirectories. Under each subdirectory, the HTML pages form cycles. A quick inspection at the pages contained in this data set reveals that there are at least thousands of possible (not necessarily disjoint) cycles, of which only ten are interesting (because they were so generated and placed in subdirectories).

We apply our SC1 and SC2 algorithms to the Python data set and ten sequences are detected by both of them. The lengths of the cycles are 9, 18, 21, 32, 64, 83, 99, 116, 120 and 836. The output sequences were manually verified to be completely accurate. Thus, we also know that SC2, which is an approximate version of SC1, fares equally well when the data sets are simple. The Python data set indeed has files with numeric suffixes, which is exactly the additional assumption that SC2 makes.

Next, we detected sequences in the WB1 and WB13 data sets using SC1 and SC2.

Table 3.2: Number of sequences detected by SC1 ($NSC1$) and SC2 ($NSC2$), and percentage of their intersection

Data Set	$NSC1$	$NSC2$	$\frac{NSC2}{NSC1}$
WB13	1368	1338	97.8%
WB1	3376	3293	97.5%

Table 3.3: Number of pages included in sequences detected by SC1 ($PSC1$) and SC2 ($PSC2$), their percentages, and the percentage of their intersection

Data Set	No. of pages (N)	$PSC1$	$\frac{PSC1}{N}$	$PSC2$	$\frac{PSC2}{N}$	$\frac{PSC2}{PSC1}$
WB13	34343	9590	27.9%	9423	27.4%	98.3%
WB1	106025	41443	39.1%	40936	38.6%	98.8%

The number of sequences detected by SC1 and SC2, and the proportion of the sequences detected by SC1 also detected by SC2 are mentioned in Table 3.2.

We observe from Table 3.2 that SC2 detects close to 98% of the sequences detected by SC1, which uses the original definition of a sequence, on both the data sets. This confirms that *Assumption 4* is not too restrictive in practice, the reason being that web page authors do name consecutive pages of a sequence using numeric or alphabetic suffixes.

Table 3.8 lists the number of pages covered by the detected sequences. The remaining pages, i.e., 70.1% and 61.4% of WB13 and WB1, respectively, are the singleton pages, which are not considered part of any sequences. We do not strictly assert that they are not part of any sequences because of two subtle reasons:

- We have decided not to detect any sequences of size two, primarily because SC1 lacks this ability by the very definition.

- The caution being exercised in not allowing abnormally long sequences results in *wrapping*. Our threshold being chosen as 25, if there is a sequence of length 26,

it would result in our algorithm finding a sequence of length 25, and the last page would be called a singleton.

Next, we examine the lengths of the detected sequences. Histograms of the lengths of detected sequences are plotted in Figs. 3.4–3.7. A logarithmic scale has been used for the Y-axis because, otherwise, the excessively huge number of singletons and small sequences would dominate the plot. Comparing Fig. 3.4 and Fig. 3.5, we find that the sequences not detected by SC2, but detected by SC1, on WB13 data set are predominantly small ones. The case of WB1 is similar, too. Again, this reveals that sequences without suggestive root and suffix combinations are rare as the length of the sequences increase.

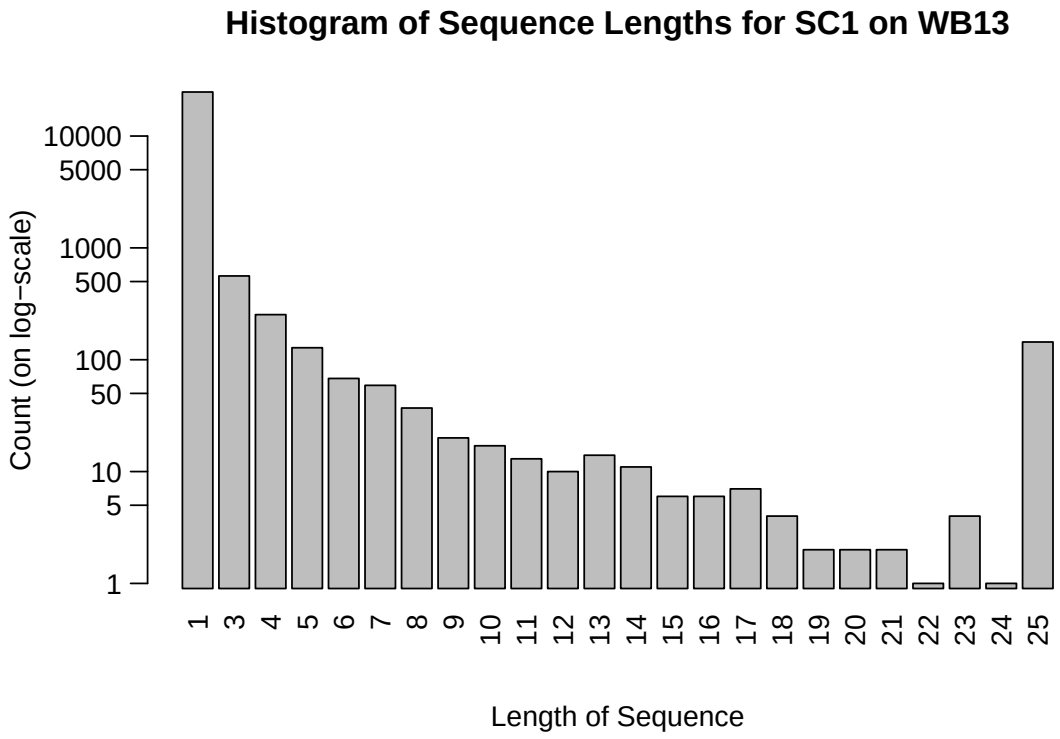


Figure 3.4: Histogram of the length of the sequences detected by SC1 on the WB13 data set

We measure the impact of merging sequences of documents on the TF and IDF of terms in the WB1 and WB13 data sets, the exact methodology being as described here.

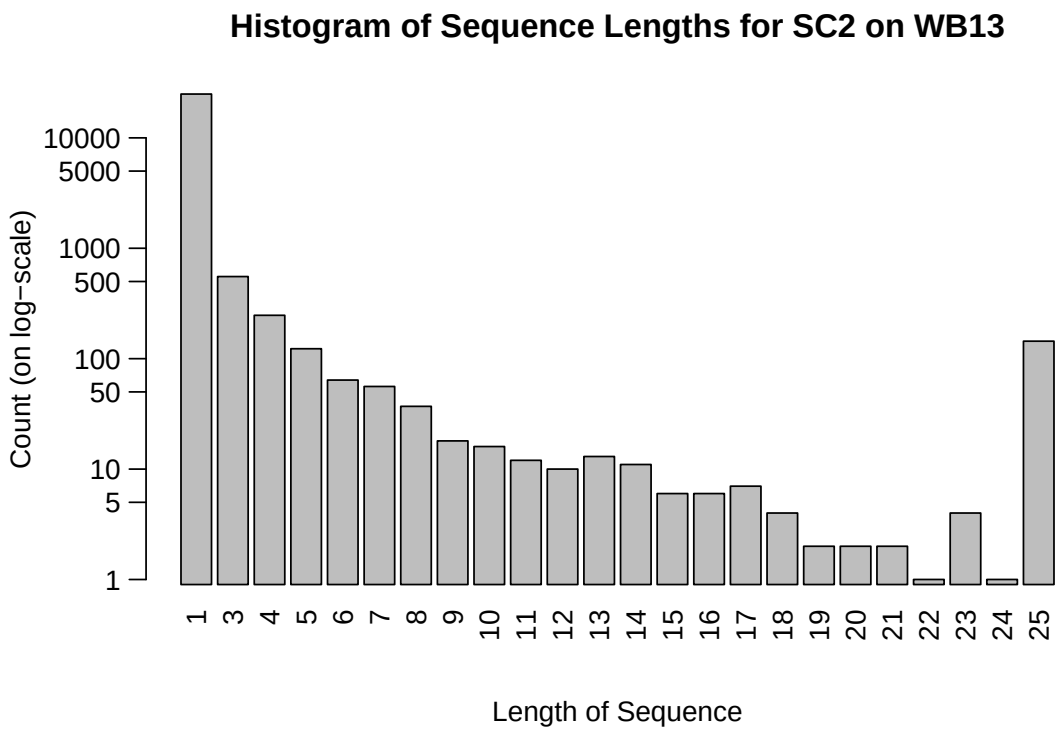


Figure 3.5: Histogram of the length of the sequences detected by SC2 on the WB13 data set

For a given corpus, we retain only words that are present in a unix dictionary (available in `/usr/share/dict/words` of most unix (especially, Linux) machines. Interestingly, this set of words includes frequently used proper nouns like Stanford and Berkeley, too. All words are folded to lower case, and are run through the Porter stemmer. Now, each document is represented as a bag of words, with the words being the aforementioned stems. Also, an inverted index is created, whereby, for each stem, the documents and the corresponding frequencies are listed. The above process is repeated for the corpus with merged sequences of documents.

Now, for each stem in the corpus (note that the set of the stems is the same in both the merged and unmerged set of documents), we compute the ratio of the IDF in the merged and the original corpus. The top gainers, in terms of IDF, for both WB13 and WB1 data

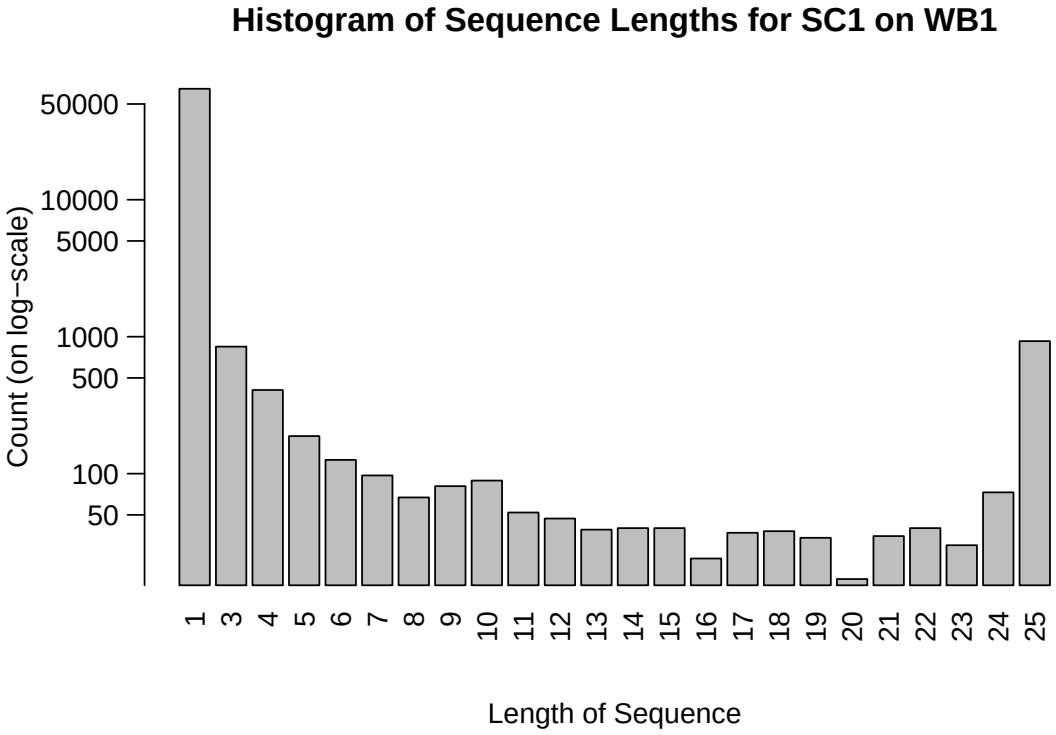


Figure 3.6: Histogram of the length of the sequences detected by SC1 on the WB1 data set

sets are shown in Figs. 3.8 and 3.9.

It is interesting to note that the words with maximum IDF gain are those that appear across several different pages of a sequence, and in a sense, are representative of the sequences they appear in. Also, interesting is the fact that stop words and other commonly used words are all pushed to the end of the list as they gain the least in terms of IDF owing to their being omnipresent in the corpora.

3.9 Conclusions

In this chapter, we have introduced a novel methodology for the task of detecting sequences of web pages. Also, the importance of sequence detection has been highlighted

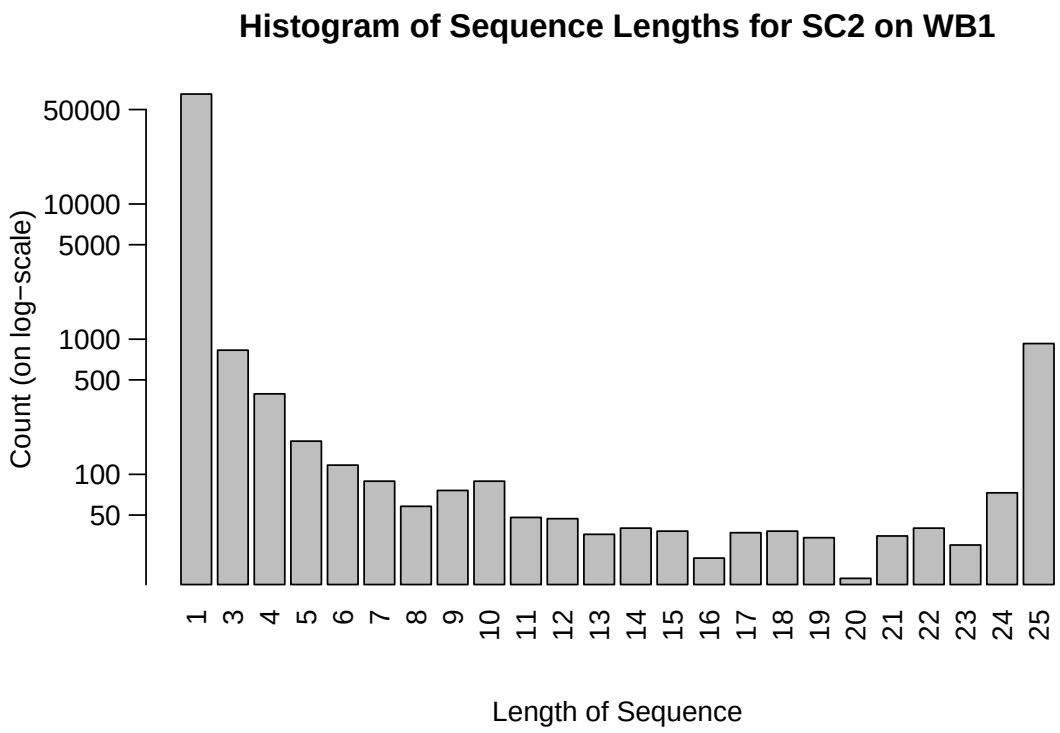


Figure 3.7: Histogram of the length of the sequences detected by SC2 on the WB1 data set

extensively. With the help of some examples, we have explained why detecting *all* possible sequences and cycles of web pages is neither feasible nor interesting. Consequently, we described the sequences of interest, and then presented a methodology for detecting only the few interesting sequences which were created to be traversed in that order. The proposed algorithms SC1 and SC2 use varying levels of domain knowledge, coded in terms of assumptions on the sequences of interest, but essentially capture the same notion that consecutive elements of a sequence have a constant relation between them. SC1 identifies continuity links in web pages, as well as, their positional information, and tracks sequences by traversing pages through links with the same positional information. SC2, on the other hand, operates directly on the URL list itself, identifying consecutive pages based on the URL strings. Experiments conducted on the Python, WB13 and WB1 cor-

pora demonstrate the effectiveness of SC1 and SC2 in detecting sequences, and also depict how merging the obtained sequences affects the term frequencies and inverse document frequences for various terms present in the corpora.

Apart from studying the problem of web page sequence detection, and providing solutions for the same, a major contribution of this chapter was to introduce novel ideas that can take advantage of sequence detection. Identifying a group of pages as being constituents of a single document reduces both the number of nodes and the number of edges in the Web graph. This, in turn, results in a reduction in the computational time and resources required for operations like page ranking on the Web graph. Given the importance that duplicate detection has received in literature, the application of sequence detection to identifying duplicates is also quite interesting. In addition, matching queries to content across multiple web pages, and thereby, returning sets of pages as web results, is a novel concept in itself.

This chapter, along with the previous one, presented work that prepares web data sets before the task of surfer modeling is taken up. In the following two chapters, we delve into the modeling process itself. In Chapter 4, we employ classical probabilistic approach while Chapter 5 incorporates the concept of fuzziness in surfer modeling.

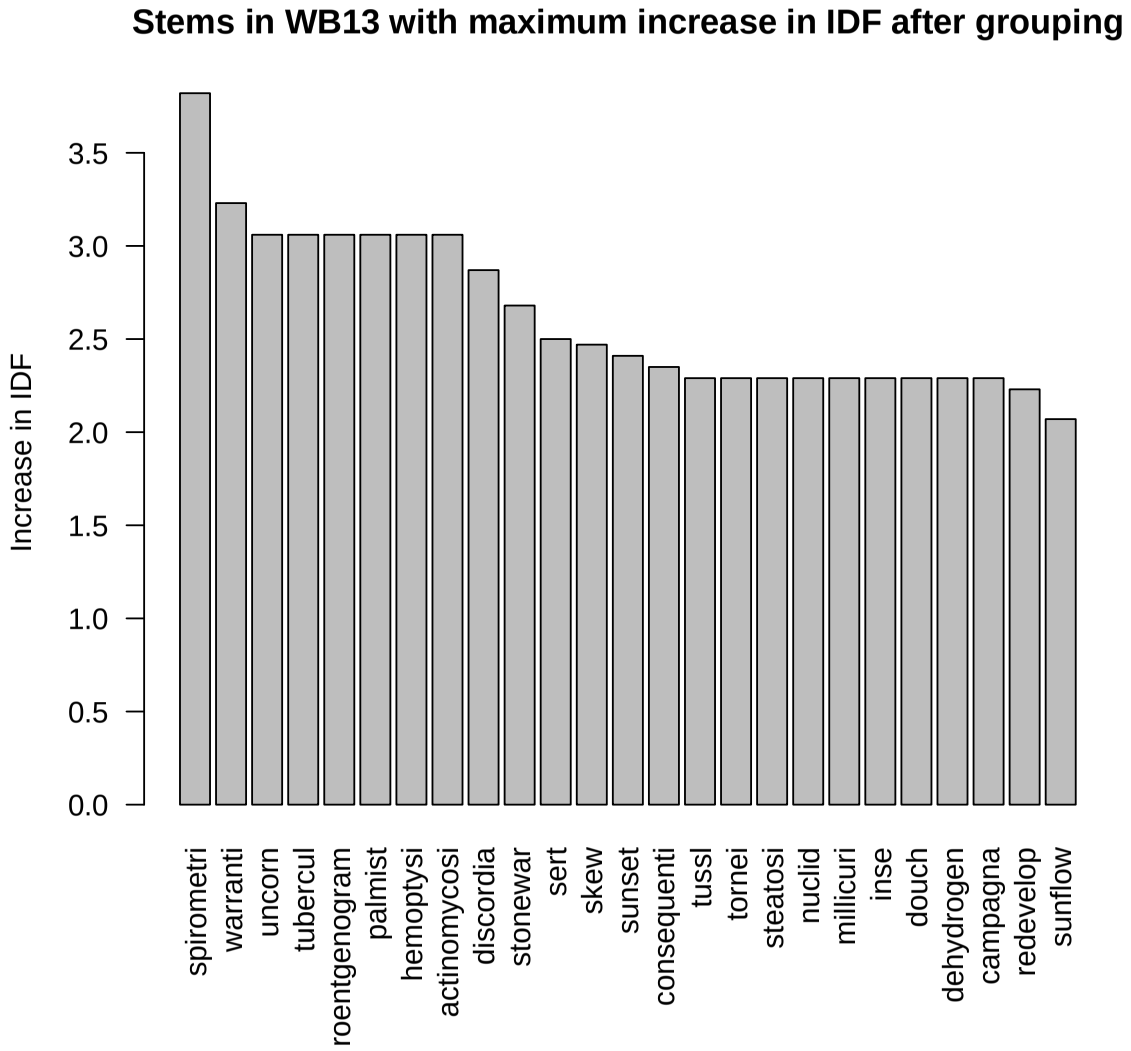


Figure 3.8: Bar plot showing the top 25 gainers in terms of IDF due to merging sequences of documents in the WB13 data set

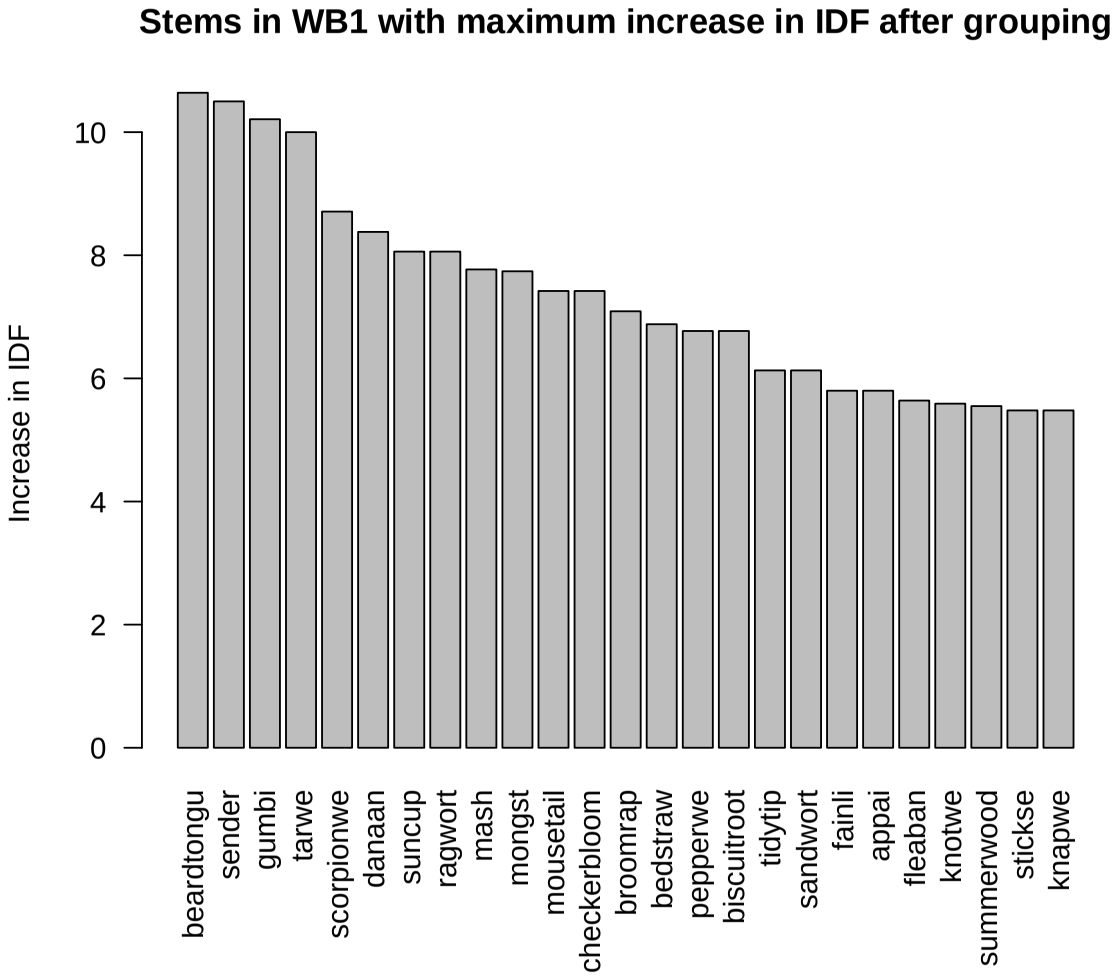


Figure 3.9: Bar plot showing the top 25 gainers in terms of IDF due to merging sequences of documents in the WB1 data set

Chapter 4

Web Surfer Model Incorporating Topic Continuity

4.1 Introduction

The present chapter deals with a methodology based on the principle of surfer models that simultaneously performs page ranking and context extraction.

Surfer models involve modeling the behavior of a user who browses the Internet. The sequence of pages that the user visits is modeled as a stochastic process $\{X_t\}$, where X_t denotes the page the surfer is on at time t. The state space for this process consists of all web documents, each page being a state that the process may attain. The transition probabilities $P(X_{t+1} = v|X_t = u)$, $1 \leq u, v \leq N$ (N is the number of pages), are defined as the probability of reaching page v in one step, given that the user is currently on page u. This transition may happen by either clicking on a link to v available in u or by typing the URL of v.

In general, one would be interested in knowing some of the properties of the process $\{X_t\}$. One such property of interest is the convergence of this process to a stationary distribution. In other words, we would like to know if, for each web page, the probability of the surfer being on that specific page converges to some value as $t \to \infty$. We may also

be interested in knowing if the above property holds regardless of the starting point of the surfer and whether it is the same value each time. These properties may be used to draw inferences about, among others, the ranks and categories of web documents.

Traditional information retrieval methods and text mining methods for the above tasks are not directly applicable to the web due to the following reasons.

The World Wide Web (WWW or the Web) is a vast network of interlinked web pages which are mostly in HyperText Markup Language (HTML) format. Other types of documents on the web are text, pdf, ps, images, *etc*. HTML documents are semistructured and contain links to other documents on the Internet. This graph structure, along with the content of all the pages, adds a new dimension to web mining over text mining.

Another distinguishing feature of the Internet is its easy access to all sorts of people. A web page may be published by anybody regardless of her profession, nationality, age, education, *etc.* Moreover, the content is not reviewed before it is made available. This leads to a severe degradation of the quality in terms of the accuracy, authenticity, integrity and consistency of the content available on the web.

Traditionally, documents were ranked on the basis of their contents. With the availabiilty of link information in web documents, and its being less prone to malicious manipulation, this information has been employed for ranking pages. The final ranking of results produced in response to a query take into consideration both the link-based ranks and the content-based ranks. It has been shown in [123] that inclusion of context in the ranking scheme greatly enhances the performance.

If the process $\{X_t\}$ has a stationary distribution, then one can actually look at the time-independent probability value $P(X = x)$, and this may be considered to be the rank of the page x. It is the unconditional probability that a surfer would be on page x.

The Random Surfer Model assumes that the surfer is browsing web pages at random by either following a link from the current page chosen at random or by typing its URL. On the contrary, the Directed Surfer Model assumes that, when the surfer is at any page, she jumps to only one of those pages that are relevant to the context, the probability of

which is proportional to the relevance of each such outlink. Both models guarantee the convergence of this stochastic process to a stationary distribution under mild assumptions like the irreducibility of the transition probability matrix. In practice, these assumptions are enforced by pruning or ignoring some links.

The present chapter is an attempt in demonstrating the significance of incorporating the information derived from another aspect, namely, the history of a user for ascertaining the transition probabilities in the surfer model for ranking a page. Here, the surfer is assumed to follow, more often than not, links on topics contained on the pages that she had visited earlier, thus maintaining a continuity of topics.

In this chapter, it is shown to be possible to simultaneously estimate both the rank and categorization of the available pages, unlike the earlier models. As a result, both categorization and ranking improve. A mathematical framework of the model is provided here along with its convergence and scalable properties. Other applications of the model, as obtained from the joint probability matrix, are also listed. The superiority of the model over some related ones is demonstrated both theoretically and experimentally on a dataset obtained from WebBase [65].

The chapter is organized as follows: the surfer models are described in detail in Section 4.2. In Section 4.3, we describe our model that incorporates the information derived from the history of a user (or, the continuity of topics) along with the motivation behind it. These are followed by the different characteristic features of the methodology and its complexity. Experimental results are given in Section 4.4, while the conclusions are drawn in Section 4.5.

4.2 Surfer Models

A variety of surfer models, such as *random surfer* [22], *HITS (Hypertext Induced Topic Selection* [73], *PHITS (Probabilistic HITS* [31], *SALSA (Stochastic Approach to Link Structure Analysis)* [86], *directed surfer* [123], *topic-sensitive pagerank* [61], *etc.*, are

available in the literature. More recently, another model called *WPSS (Web Page Scoring Systems)* has been proposed in [42]. This model is very general and each of the above mentioned models becomes a special case of WPSS.

All the above models consider random walks on the web and the rank of a page is computed as the probability of being on that page during the random walk. The model considered by WPSS [42] allows for random walks where the surfer is allowed to do one of the following: follow a forward link, go to a backward link, jump to another URL or stay in the present page. It thereby encompasses all the actions allowed by the other models. Among these models, HITS, PHITS and SALSA incorporate random walks in both forward and the backward directions. In that sense, they do not model a realistic surfer who would not know all the pages that lead to the page she currently is on. On the other hand, random surfer model, directed surfer model and topic-sensitive pagerank incorporate following only forward links, and not backward links. In the present chapter, since we are concerned about incorporating the information contained in the history of a surfer, we explain only the latter ones in some detail.

4.2.1 Random Surfer Model

The random surfer model models a user who keeps visiting new pages by clicking, at random, the links available on the current page. Thus, given that the surfer is on page v at time t, the probability of her being on page u at time $t+1$, $P(X_{t+1} = u|X_t = v)$, is assumed to be $\frac{1}{|F_v|}$, where F_v is the set of forward links from v. Therefore, the probability of the surfer being on page u at time $t+1$ is computed as

$$P(X_{t+1} = u) = \sum_{v=1}^{N} P(X_{t+1} = u|X_t = v)P(X_t = v), \tag{4.1}$$

where N is the total number of pages. Counting only those pages v for which $P(X_{t+1} = u|X_t = v) > 0$ (i.e., those pages which have a link to u), we have

$$P(X_{t+1} = u) \tag{4.2}$$

$$= \sum_{v \in B_u} P(X_{t+1} = u | X_t = v) P(X_t = v) \tag{4.3}$$

$$= \sum_{v \in B_u} \frac{P(X_t = v)}{|F_v|}, \tag{4.4}$$

where B_u denotes the set of backlinks of u. Here, the second equality is a consequence of assuming that an outgoing link would be chosen at random.

Let the transition matrix for the stochastic process $\{X_t\}$ be denoted by

$$M = ((m_{uv}))_{u,v \in \{1,2,\ldots,N\}}. \tag{4.5}$$

We then have,

$$m_{uv} = \frac{l_{uv}}{\sum_{w=1}^{N} l_{wv}}, \tag{4.6}$$

where l_{uv} denotes the (u, v)th element of the link matrix of the web graph, and is defined as being equal to 1 if and only if v has a link to u, for $u, v \in \{1, 2, \ldots, N\}$. Let R_u^t denote $P(X_t = u)$. Then, the probability distribution of X_{t+1}, $\mathbf{R}^{(t+1)} = (R_1^{t+1}, R_2^{t+1}, \ldots, R_N^{t+1})^T$, may be recursively defined as,

$$\mathbf{R}^{(t+1)} = M\mathbf{R}^{(t)}. \tag{4.7}$$

If this stochastic process has a stationary distribution, it would satisfy $\mathbf{R} = M\mathbf{R}$. The uth element of the vector $\mathbf{R}$ is the unconditional probability of the surfer being on page u, and may be considered to be the rank of page u, and $\mathbf{R}$ may be called the rank vector. To compute $\mathbf{R}$, which is nothing but the dominant (or principal) eigenvector of M, the power method is employed, whereby $\mathbf{R}^t$ converges to $\mathbf{R}$ as $t \to \infty$. This is the basic idea behind the PageRank algorithm suggested by Brin and Page [22].

This manner of page ranking is quite similar to employing manual reviewers. In the case of manual reviewing, a few selected/qualified people review web pages or sites submitted for reviewing. PageRank, on the other hand, considers all the authors of web pages as its reviewers and a reviewer rates a web page highly by providing a link to that page from her own page.

Now, the process $\{X_t\}$, defined in the above manner, may have certain absorbing sets of states (which make the process reducible). In terms of web pages, this means that certain sets of web pages may not have links to pages outside them. These are known as rank sinks (or leaks when the sets are singletons). Under the assumed model, the surfer is bound to be stuck in one of these sets of pages as t increases. In practice, however, this is not reasonable, as the surfer can visit a page outside an absorbing set of pages, by typing its URL. To reflect this, the model is modified slightly. It is assumed that when the surfer is on a page v, he may either decide to type the URL of a new page, the probability of which is taken to be d ($d > 0$),or follow one of the links available on the page. That is,

$$m'_{uv} := P(X_{t+1} = u | X_t = v) = \frac{d}{N} + \frac{1-d}{|F_v|}. \tag{4.8}$$

This makes the stochastic process irreducible and the corresponding web graph strongly connected. The PageRank vector can now be computed as the principal eigenvector of $M' = ((m'_{uv}))$. In practice, pages having no outlinks are kept out of the PageRank computation, and are plugged in later.

The order of the matrix M' may be very large, sometimes running into several billion, and the computation of PageRank needs enormous effort. Consequently, efficient schemes considering both time and space requirements have been reported, [60, 70].

4.2.2 Directed surfer model

Richardson and Domingos [123] have modeled a more intelligent surfer, who probabilistically chooses the next page to be visited depending on the content of the pages and the query terms the surfer is looking for. The transition probabilities are calculated in terms of a relevance function $R_q(u)$ that computes the relevance of page u to query q. This is an extension of the one-level influence propagation model introduced in [119].

The probability of reaching page u from v, m'_{uv}, is computed as

$$m'_{uv} = (1-\beta)P'_q(u) + \beta P_q(v \to u), \tag{4.9}$$

where, $P'_q(u)$ and $P_q(v \rightarrow u)$ are arbitrary distributions. $P'_q(u)$ is the probability that the surfer reaches page u (without following a link) in the context of q and corresponds to the bias vector in the PageRank computation. $P_q(v \rightarrow u)$, on the other hand, is the probability of choosing u in the context of q from among the links provided on page u. In practice, $P'_q(u)$ and $P_q(v \rightarrow u)$ may be derived from a relevance measure as

$$P'_q(u) = \frac{R_q(u)}{\sum_{v=1}^{N} R_q(v)}, \tag{4.10}$$

and

$$P_q(v \rightarrow u) = \frac{R_q(u)}{\sum_{z \in F_v} R_q(z)}, \tag{4.11}$$

where, $R_q(u)$ is the relevance of u to q. The rank vector computed in this manner is termed *Query Dependent PageRank*, or QD-PageRank, in short.

The choice of the relevance function is arbitrary. If $R_q(u) = 1, \forall\, q, u$, it is the random surfer model. Other suggestions for $R_q(u)$ provided in [123] include an indicator function for the presence of q in u and TFIDF-like scores for q in u. (TFIDF stands for Term Frequency-Inverse Document Frequency). The latter ones make the model more efficient. In the experiments reported in [123], $R_q(u)$ was chosen to be the fraction of words equal to q in the page u.

Even though it is not explicitly stated in [123], the bias vector and the transition probabilities may be obtained from different relevance measures (say, R'_q and R_q, respectively). In this connection, Haveliwala's recent work on topic-sensitive ranking [61, 62] of web pages may be mentioned, where a PageRank vector is computed for each distinct topic. For each topic, a different bias vector is used during the computation of PageRank, where the bias vector contains non-zero entries corresponding to only those pages which appear under that particular category in ODP directory. The topic is decided on the basis of the context of the query.

This investigation [61, 62] may, therefore, be treated as a special case of the directed surfer model where the values assumed by q are from the set of categories listed under the Open Directory (at `http://dmoz.org`). In this case, the choice of R'_q and R_q is as

follows:

$$R'_q(u) = \mathcal{I}[u \text{ appears under category } q \text{ in ODP}], \tag{4.12}$$

where, $\mathcal{I}$ is the indicator function and

$$R_q(u) = 1 \; \forall \; q, u. \tag{4.13}$$

Note that although this algorithm does not make use of the content in the individual pages for deciding upon the transition probabilities for choosing one of the outlinks, this aspect can be incorporated, as in [123].

The improvement of performance in page ranking due to incorporation of the said content can be explained through an example in Fig. 4.1.

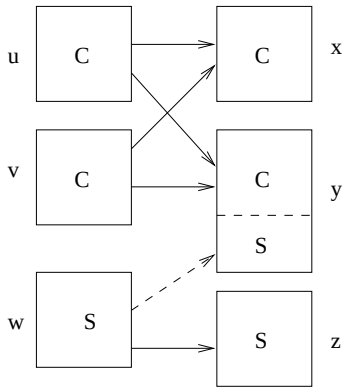


Figure 4.1: Example 1: showing the significance of page content

In Fig. 4.1, we have a set of web pages on two topics, e.g., *computers* (C) and *sports* (S). Let us assume that pages x and y have similar content on topic C but x is more relevant to a query q on topic C than y. Let us also assume that the content on topic S is not relevant to q.

In response to the query q, the directed surfer model computes the query dependent ranks as follows: x is more relevant than y for q and, so, $P_q(u \rightarrow x) > P_q(u \rightarrow y)$ and $P_q(v \rightarrow x) > P_q(v \rightarrow y)$. Since, S is not relevant to q, $P_q(\cdot \rightarrow w)$, the probabilty of reaching w from any of its backlinks, is zero and consequently, $Rank_q(w) = 0$. Now,

$$Rank_q(x) = P(u \rightarrow x) * Rank_q(u) + P(v \rightarrow x) * Rank_q(v) \tag{4.14}$$

and,

$$Rank_q(y) = P(u \rightarrow y) * Rank_q(u) + P(v \rightarrow y) * Rank_q(v). \quad (4.15)$$

Thus, $Rank_q(x) > Rank_q(y)$, and x will appear before y in the results for q, as desired.

Haveliwala's algorithm, on the other hand, computes the ranks in the following manner: At first, the topic of q is determined based on text analysis. In the present case, it is C. So C-sensitive PageRank ($Rank_C$), where the bias vector consists of non-zero entries for only pages that appear under the category C of Open Directory [109], is used for the purpose of ranking the results. The C-sensitive ranks for x and y are computed as

$$Rank_C(x) = 0.5 * Rank_C(u) + 0.5 * Rank_C(v) \quad (4.16)$$

and,

$$Rank_C(y) = 0.5 * Rank_C(u) + 0.5 * Rank_C(v) + 0.5 * Rank_C(w). \quad (4.17)$$

Now, even though w has content only on S, $Rank_C(w)$ need not be zero due to *topic drift* [27, 28], where a sequence of links followed by the surfer may lead him onto a completely different topic than what she started with. Thus, $Rank_C(y)$ turns out to be greater than $Rank_C(x)$, which is not appropriate. Therefore, despite creating a topic sensitive bias vector, it seems reasonable also to incorporate the content information for computing the transition probabilities, thereby reducing the value of $Rank_C(y)$.

4.3 Surfer Model Incorporating History

4.3.1 Motivation

We illustrate here the need for a surfer model which can incorporate the history of the surfer. Let us consider Fig. 4.2 and let page *w* be the page currently being browsed by a user. This page contains content on two distinct topics, namely, C and S. It is desired to compute the probabilities with which the user moves on to *x*, *y* and *z*, respectively by

clicking on a link on page *w*. According to the random surfer model, each of the above probabilities equals $\frac{1}{3}$.

The directed surfer model uses the similarity of the contents (of *x*, *y* and *z* with *w*) for computing the transition probabilities. In this case, by virtue of their contents, the similarities of *x*, *y* and *z* with *w* are approximately equal, thereby making the transition probabilities again to be approximately $\frac{1}{3}$ each.

It may be noted that, under both the random and directed surfer models, these transition probabilities remain the same irrespective of whether the surfer was at u or v prior to reaching w. In case the surfer was at u, intuitively, it is more likely that she would move on to x instead of y or z. In other words, the notion of the surfer's history affecting the choice of outlink may be useful in determining the link that the user follows.

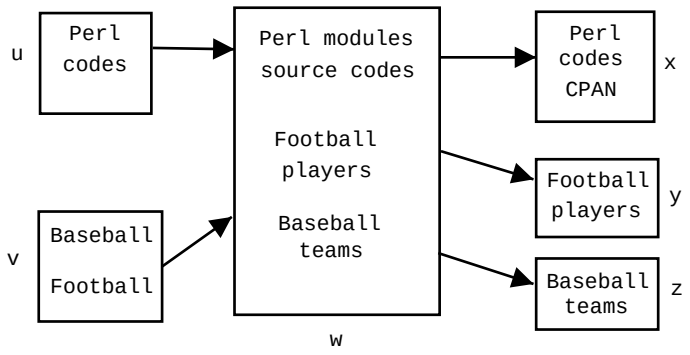


Figure 4.2: Example 2 showing the significance of surfer history

4.3.2 Theory

As seen in the above example, it is evident that the transition probabilities depend on the pages visited prior to reaching the current page. In order to incorporate this dependency, we propose a new surfer model, where a surfer moves on to pages that match his topic of interest. We assume that every page may have content on one or more topics and the user chooses one of them as her topic of interest. Usually, the surfer moves on to a new

page in keeping with her topic of interest. However, occasionally she may also visit other pages, say, out of curiosity.

Even though one might be tempted to conclude that the topic of interest may be estimated from the page's contents itself, Example 4.2 proves the contrary. The topic of interest is guessed by looking at the pages from which the page under consideration is reached. The knowledge of pages visited previously may be utilized by an online algorithm that computes the transition probabilities each time the user visits a new page.

Our primary interest being in offline applications, we guess, probabilistically, the history of the surfer, and thereby estimate the topic of interest. We compute a set of transition probabilities under the assumption that a user generally browses with a particular topic of interest and is more likely to browse pages on similar topics rather than dissimilar ones. We formally introduce our model as follows:

Topic Continuity Model

Let X_t and I_t denote the page the surfer is on and her topic of interest, respectively, at time t. We assume that any surfer is interested in one of T distinct topics. We also assume that the probability of the surfer changing her topic of interest at any given time is ϵ ($\epsilon < 0.5$), i.e.,

$$P\left(I_{t+1} \neq k | I_t = k, X_t = v\right) = \epsilon, \forall t, k \text{ and } v. \tag{4.18}$$

Note that this implies that

$$\begin{aligned} & P\left(I_{t+1} \neq I_t\right) \\ &= \sum_k P\left(I_{t+1} \neq k | I_t = k\right) P(I_t = k) \\ &= \sum_k \sum_v P\left(I_{t+1} \neq k | I_t = k, X_t = v\right) P(I_t = k, X_t = v) \\ &= \epsilon \end{aligned}$$

Further, we assume that any of the $(T-1)$ remaining topics are equally likely to be

chosen when a change does happen, that is,

$$P\left(I_{t+1}=k|I_t=l,X_t=v\right)=\begin{cases}(1-\epsilon) & \text{if } k=l\\ \frac{\epsilon}{T-1} & o.w.\end{cases} \tag{4.19}$$

We can expand the probability of the state at time $(t+1)$ in terms of the probability conditioned on the knowledge of the state at time t. So, the joint probability of the surfer being on a page z and her topic of interest being k at time $(t+1)$ may be written as follows:

$$\begin{aligned}
&P(I_{t+1}=k,X_{t+1}=z)\\
&= \sum_{v\in B_z}\sum_{l} P(I_{t+1}=k,X_{t+1}=z|I_t=l,X_t=v)P(X_t=v,I_t=l)\\
&= \sum_{v\in B_z}\sum_{l} P(X_{t+1}=z|I_{t+1}=k,I_t=l,X_t=v)*\\
&\quad P(I_{t+1}=k|I_t=l,X_t=v)P(I_t=l,X_t=v).
\end{aligned} \tag{4.20}$$

Now, substituting ϵ or $1-\epsilon$, as the case may be, and rearranging the terms, we have

$$\begin{aligned}
&P(I_{t+1}=k,X_{t+1}=z)\\
&= \sum_{v\in B_z}(1-\epsilon)P(X_{t+1}=z|I_{t+1}=k,X_t=v)P(I_t=k,X_t=v)\\
&\quad +\sum_{v\in B_z}\frac{\epsilon}{T-1}\sum_{l\neq k}\{P(X_{t+1}=z|I_{t+1}=k,X_t=v)P(I_t=l,X_t=v)\}\\
&= \sum_{v\in B_z}(1-\epsilon)P(X_{t+1}=z|I_{t+1}=k,X_t=v)P(I_t=k,X_t=v)\\
&\quad +\sum_{v\in B_z}\frac{\epsilon}{T-1}P(X_{t+1}=z|I_{t+1}=k,X_t=v)*\\
&\quad \left\{\sum_{l}P(I_t=l,X_t=v)-P(I_t=k,X_t=v)\right\}\\
&= \sum_{v\in B_z}P(X_{t+1}=z|I_{t+1}=k,X_t=v)*\\
&\quad \left\{\left(1-\frac{\epsilon T}{T-1}\right)P(I_t=k,X_t=v)+\frac{\epsilon}{T-1}\sum_{l}P(I_t=l,X_t=v)\right\}.
\end{aligned} \tag{4.21}$$

From Eq. 4.21 it is clear that the proposed surfer model satisfies the Markov property, in the sense that given the state at time t, the surfer's behavior at time $(t+1)$ is independent

of that at time $(t-1)$ (and anything prior to that). It may be noted that the history of the surfer has already been implicitly taken into account through the probabilities of each achievable state of the surfer at time t.

Now, as in the Directed Surfer Model, we assume that

$$P(X_{t+1} = z | I_{t+1} = k, X_t = v) = \frac{P(I_t = k | X_t = z)}{\sum_{y \in F_v} P(I_t = k | X_t = y)}, \tag{4.22}$$

the difference being that these probabilities are now time-dependent, whereas in the Directed Surfer Model [123], they are not. Plugging this into Eq. 4.21, we have

$$\begin{aligned} & P(I_{t+1} = k, X_{t+1} = z) \\ & = \sum_{v \in B_z} \frac{P(I_t = k | X_t = z)}{\sum_{y \in F_v} P(I_t = k | X_t = y)} \left\{ \left(1 - \frac{\epsilon T}{T-1}\right) P(I_t = k, X_t = v) \right. \\ & \quad \left. + \frac{\epsilon}{T-1} P(X_t = v) \right\} \end{aligned} \tag{4.23}$$

Note that, in Eq. 4.23, if we substitute $P(I_t = k, X_t = z)$ by $\frac{1}{T} P(X_t = z)$, (i.e., any topic is equally likely on any given page), the equation simplifies to

$$\frac{1}{T} P(X_{t+1} = z) = \frac{1}{T} \sum_{v \in B_z} \frac{1}{|F_v|} P(X_t = v),$$

which is the same as the equation for the random surfer model.

Similarly, if we had set ϵ to be 0, choosing

$$P(X_{t+1} = z | I_{t+1} = k, X_t = v) = \frac{P(I_0 = k | X_0 = z)}{\sum_{y \in F_v} P(I_0 = k | X_0 = y)}, \forall\, t,$$

Eq. 4.21 is similar to the Directed Surfer Model, with the random jump factor excluded.

Although, the present work generalizes existing surfer models, the topic continuity model is non-linear, and therefore, theoretically proving that $P(I_t = k, X_t = v)$ converges with t for all values of v and k has not been possible. In practice, however, the method has been experimentally observed to converge, details of which are provided in Section 4.4.3.

4.3.3 Obtaining initial estimates

For faster convergence of the above iterative procedure, a good initial estimate of the joint probability distribution is necessary. The joint probability $P(I_0, X_0)$ is estimated as $P(I_0|X_0)P(X_0)$, where $P(X_0)$ is obtained using an existing version of PageRank. The quantities $P(I_0|X_0)$ have been estimated using the Naive Bayes algorithm [88], a standard method for text classification, as $P(I_0 = C_j|X_0) \propto P(C_j)\prod_{i=1}^{K} P(x_i|C_j)$ (the denominator, $P(X_0)$, is common), where the document X_0 is treated as the term vector $(x_1, x_2, \ldots, x_K)$, and C_j is the jth topic listed under the Open Directory. The topic-

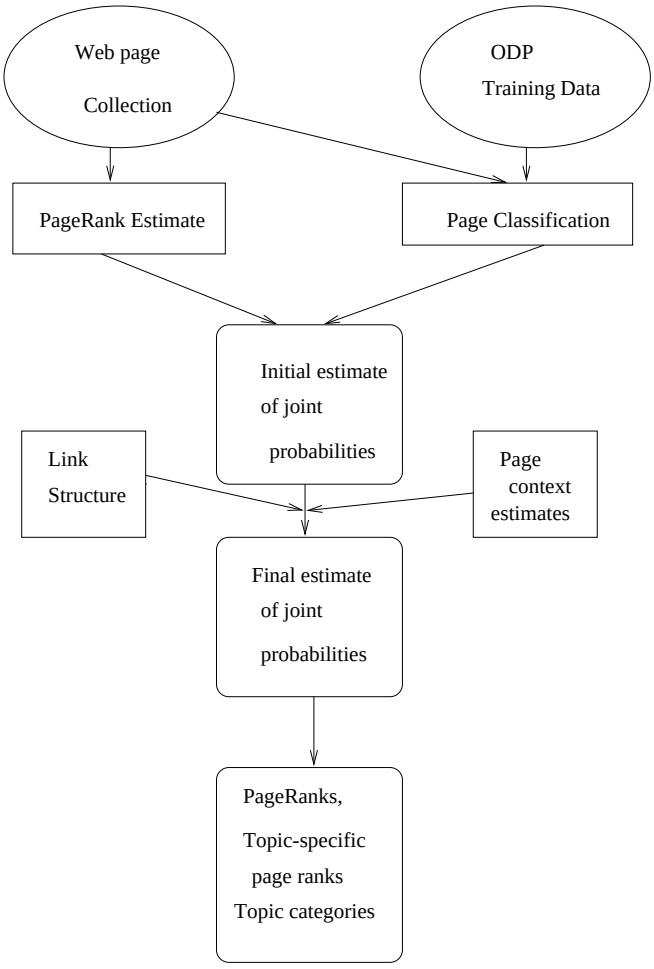


Figure 4.3: Flowchart for the Topic Continuity Model

conditional probabilities for each term, $P(x_i|C_j)$, and the prior probabilities of the topics $P(C_j)$, are estimated as the corresponding frequencies obtained from the pages listed under the Open Directory.

The vector $(P(I_0 = C_j|X_0))_{j=1,2,\ldots,K}$ is then normalized and its jth element is the probability of the page having content on jth topic. It may be noted that this need not be the desired conditional probability value that we intend to estimate ultimately. For example, a page may appear quite relevant to the topic, say "Business", however, its primary topic may be something different, say "News", depending on the context (or neighborhood) of the web page. Nevertheless, this classification suffices to serve our purpose of finding an initial estimate of the matrix G.

These initial estimates are then plugged into the iterative procedure. For ease of understanding, we provide a (simplified) flowchart depicting the steps involved in the proposed algorithm.

4.3.4 Page Ranking, Categorization, and Other Uses

As described above, we have obtained the stationary (joint) distribution of (I, X). The probability matrix may be written as:

$$G = \begin{matrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ g_{K1} & g_{K2} & \cdots & g_{KN} \end{matrix} \tag{4.24}$$

The rows and columns stand for topics and web pages, respectively. Each element of this matrix, g_{kv}, represents the joint probability of a topic of interest and a web page, $P(I = k, X = v)$. We note the following properties of this matrix, along with their utilities.

- The sum of the vth column,

$$\sum_{k=1}^{K} g_{kv} = g_{.v} = P(X = v), \tag{4.25}$$

is the rank of page v. So the quantity $g_{.v}$ may be used for *unconditionally ranking* web pages.

- The sum of the kth row,

$$\sum_{v=1}^{N} g_{kv} = g_{k.} = P(I = k), \tag{4.26}$$

is the unconditional probability of a surfer's topic of interest being k. This has significance in obtaining *topic representations on the web*.

- The (v, k)th element divided by its row total,

$$\frac{g_{kv}}{g_{k.}} = \frac{P(I = k, X = v)}{P(I = k)} = P(X = v|I = k), \tag{4.27}$$

is the *topic specific page rank* of page v (for topic k). In other words, g_{kv} is proportional to the rank of v for topic k.

- The (k, v)th element divided by its column total,

$$\frac{g_{kv}}{g_{.v}} = \frac{P(I = k, X = v)}{P(X = v)} = P(I = k|X = v), \tag{4.28}$$

is the probability of the topic of interest being k when the surfer is on page v and hence, the vth column provides the *relevance of a web page* on each of the K topics.

- The parameter ϵ, as mentioned in Section 4.3.2, controls the curiosity factor of the surfer. The higher the ϵ, the more curious she is. In other words, the lower the ϵ, the more focused is the surfer and less frequently tends to change the topics of interest. This can be applied to *personalization* where the behaviour of users is quite varied.

Note that the proposed algorithm modifies PageRank exactly in the same way as PHITS [31] modifies HITS (Hypertext Induced Topic Selection) [73]. HITS like algorithms are prone to link spamming, where, to improve the authority a new page, all that one needs to do is create several pages that link to existing authoritative pages as well as

the new one. On the other hand, the proposed algorithm inherits its robustness from the PageRank algorithm.

Earlier investigations had generally, focused on either page ranking or page categorization. What the current investigation does is to perform both these inter-dependent tasks simultaneously. This notion had been mentioned in [36] and had also been independently reported in a preliminary form in [101].

It is interesting to note that the proposed algorithm does not actually categorize, in its true sense, a web page's contents. It just estimates what a surfer would be interested in when she reaches a page. This means that even though the terms appearing in a document are suggestive of some particular category, the topics and ranks of the pages linking to it play a major role in determining if it indeed is relevant for that category. The scores that we compute for each page can be considered equivalent to categorization in the sense that this is what a user visiting this page would think about.

We mention here that though both Haveliwala's topic-sensitive PageRank algorithm and the proposed one make use of the topic information available under the ODP directory, there is a difference in the manner in which it is employed. While the former one needs information about which topics a URL is listed under, the latter needs some text categorization mechanism, which in this particular case, happens to be derived from the ODP data. Since Haveliwala's algorithm does not need to categorize each available page, it is computationally more efficient compared to our algorithm. However, by virtue of this extra effort, the proposed methodology counters topic drift by controlling the transfer of rank between dissimilar pages.

4.3.5 Complexity and scalability

We now discuss the complexities involved in the proposed algorithm. Let K be the number of topics under consideration. Then the disk space required is K times that required for ordinary PageRank and is the same as that for topic-sensitive PageRank [61]. As noted in [123], despite being higher than the requirements for the random surfer model, it is far

smaller than that for the index (of the original page contents) itself. The time complexity for the computation of the matrix G is as follows: From Eq. 4.23, it is obvious that, in each iteration and for each k and z, the computation of g_{kz} needs $\mathcal{O}(|B_z|)$ computations (corresponding to each backlink v of z). Thus, for all the KN entries in G, letting $\bar{B}$ denote the average number of backlinks of a page, $\mathcal{O}(NK\bar{B})$ computations are required during each iteration. This is about the same as that for topic-sensitive PageRank, although our algorithm involves some additional overhead for computing $P(X = v)$ and $\sum_{y \in F_v} P(I = k|X = y)$ at the end of each iteration. Note that we do not count the preprocessing steps like stemming, stopword removal and creation of an inverted index as they are the same for any such algorithm.

4.4 Experimental Results

The performance of the proposed methodology has been evaluated along with comparisons with some of the existing algorithms. Here we discuss the data sets used, and the methods of implementation and evaluation.

4.4.1 Data Sets Used

A training data set is obtained from the Open Directory Project, which is the largest, most comprehensive human-edited directory of the Web. It is constructed and maintained by a vast, global community of volunteer editors [109]. A file in RDF (Resource Description Framework) format has been downloaded from the Open Directory [109]. This file consists of URLs and their description organized into seventeen distinct topics (Table 4.1). The words available in the description are assumed to represent, to some extent, the topics under which they appear.

The test data set, consisting of approximately five million pages was obtained from WebBase [65] (more could not be obtained due to local constraints on disk space). These pages form a connected neighborhood of the web. We have used a stream based access

Table 4.1: Top level categories available in the RDF file obtained from ODP

Adult	Recreation
Arts	Reference
Business	Regional
Computers	Science
Games	Shopping
Health	Society
Home	Sports
Kids_And_Teens	World
News	

to download pages, whereby pages are retrieved in the order they were crawled [65]. The original pages contained raw HTML along with a header that consisted of the page's URL, timestamp, *etc.* Only the URL information was used in our experiments. Words that were obtained from the above mentioned RDF file were the only ones that were retained and the rest were discarded. The links were normalized and self-links were removed. The links were stored separately and an inverted index was created for the downloaded collection. Only the counts of the words were stored, as is done in the vector space model.

Note that the ODP data set has information on the categories of pages, but no link information. On the other hand, the WebBase data set has link information but no categorization. That is why we used ODP data for training based on the categorization information, while the text and link information in the WebBase data is used for testing.

For estimating the value of ϵ, we have used the *msnbc.com anonymous web data* (available at `http://kdd.ics.uci.edu/databases/msnbc/msnbc.html`), which has transition information between categories unlike the ODP and WebBase data sets. This data describes the page visits of users who visited msnbc.com on September 28, 1999. Visits are recorded in time order and only the category of the page requested is

stored. There are seventeen categories (Table 4.2).

Table 4.2: Categories in the MSNBC data set

frontpage	living
news	business
tech	sports
local	summary
opinion	bbs
on-air	travel
misc	msn-news
weather	msn-sports
health	

4.4.2 Implementation

During training, initial page categorizations were obtained using the Naive Bayes algorithm, as already mentioned.

For estimating the value of ϵ, it is observed that of a total of 3708976 transitions during 4698820 visits in the msnbc data, over 65% transitions were between pages on the same category, i.e., there were about 35% cross-topic transitions. Accordingly, we chose the value of ϵ to be 0.35. Note that, this data set did not capture any requests that would have been served from the user's cache. Had these requests been included in the data set, the number of within-transitions would have been higher, i.e., ϵ value would have been lower. In order to reflect this, we also conducted experiment for a lower value of $\epsilon = 0.2$, to reflect a more focused surfer.

For an efficient implementation of the page rank computation, we rewrite Eq. 4.23 as:

$$P(I_{t+1} = k, X_{t+1} = z)$$

$$= \frac{P(I_t = k, X_t = z)}{P(X_t = z)} \sum_{v \in B_z} \frac{\left\{\left(1 - \frac{\epsilon T}{T-1}\right) P(I_t = k, X_t = v) + \frac{\epsilon}{T-1} P(X_t = v)\right\}}{\sum_{y \in F_v} P(I_t = k | X_t = y)} \tag{4.29}$$

At the end of each iteration, we compute the following for each page v:

$$P(X_t = v) = \sum_k P(I_t = k, X_t = v), \tag{4.30}$$

and

$$\sum_{y \in F_v} P(I_t = k | X_t = y) = \sum_{y \in F_v} \frac{P(I_t = k, X_t = y)}{P(X_t = y)}, , \tag{4.31}$$

While traversing the adjacency matrix of the web graph,

$$\sum_{v \in B_z} \frac{\left\{\left(1 - \frac{\epsilon T}{T-1}\right) P(I_t = k, X_t = v) + \frac{\epsilon}{T-1} P(X_t = v)\right\}}{\sum_{y \in F_v} P(I_t = k | X_t = y)} \tag{4.32}$$

is computed separately. At the end of the iteration, we multiply the above quantity by $\frac{P(I_t=k, X_t=z)}{P(X_t=z)}$, thus obtaining the value for $P(I_{t+1} = k, X_{t+1} = z)$, for each $1 \leq k \leq K$, $1 \leq z \leq N$.

The iterations were allowed to run until convergence in terms of the L_∞ norm. For improved precision during computations, we had scaled up the whole joint probability matrix, $((P_{kz}))_{1 \leq k \leq K, 1 \leq z \leq N}$ by N (so that the sum of each column of the matrix would be 1, on the average). In other words, the procedure was stopped as soon as

$$\max_k \max_z \left(N * P(I_{t+1} = k, X_{t+1} = z) - N * P(I_t = k, X_t = z)\right)$$

was less than a threshold. In our experiments, this threshold was chosen to be 0.001.

4.4.3 Evaluation

The process of evaluation consists of two parts: the first part deals with comparison of the ranks of the pages and the second with their contexts.

The ranks obtained by our method were compared against those obtained by PageRank [22] and QD-PageRank [123]. Five volunteers were chosen for this purpose. Ten queries

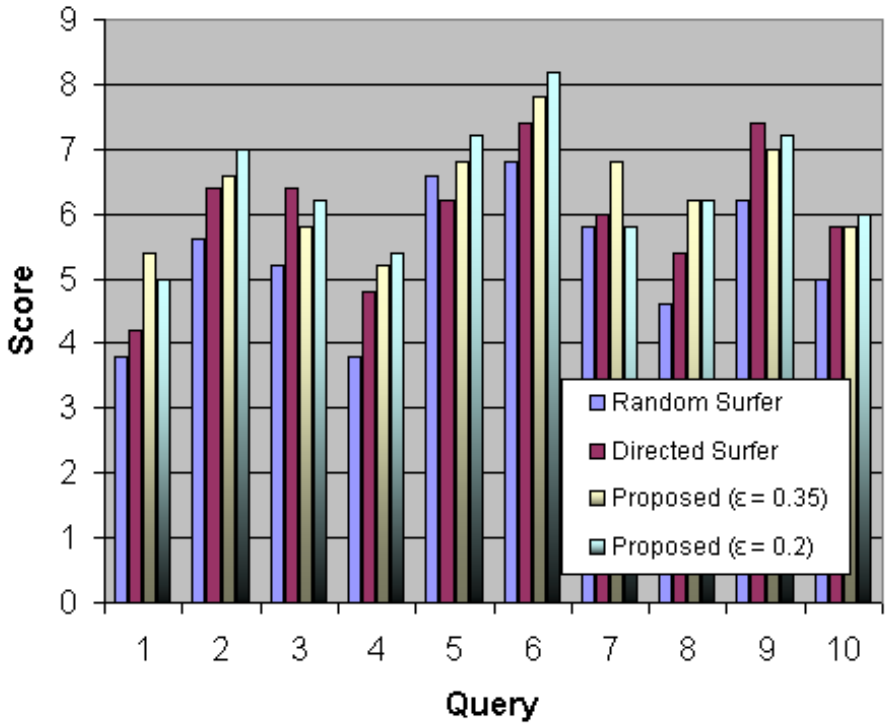


Figure 4.4: Page Rank Comparison

were chosen from those used in [61] (some queries were modified considering the size of the collection available with us).

The queries used for evaluation are provided in Table 4.3. The top ten pages obtained in response to the queries by the four algorithms, namely, random surfer model (or PageRank), directed surfer model (or QD-PageRank), proposed approach with $\epsilon = 0.35$ and proposed approach with $\epsilon = 0.2$, were studied by the five volunteers. They provided a rating (or score) between 0 and 10, a rating of 10 being the best, to each algorithm for each query. The average values obtained for each query are presented in Fig. 4.4. We have performed pairwise comparisons testing for difference of the means using a t-test with 9 degrees of freedom. The null hypothesis was taken to be that the means were equal, while the alternate hypothesis was that the second method (the one appearing later in the bar-plot) fared better. The tests gave the following results:

- The proposed method (for both the above mentioned values of ϵ) and the directed surfer model significantly outperform the random surfer model at a confidence level of 95%.

- The scores obtained by the proposed method with $\epsilon = 0.2$ show a significant improvement over those obtained by the directed surfer model at a confidence level of 95%.

- The improvement in scores over the directed surfer model obtained by the proposed method with $\epsilon = 0.35$ is significant at a confidence level of 90%, but not at a confidence level of 95%.

- No significant difference was observed (that is, the null hypothesis was accepted) between the scores of the proposed algorithm for the two choices of ϵ.

These findings further strengthen the theoretical observations as mentioned in Section 4.3.4.

Table 4.3: Queries used for comparing page ranks

1	architecture
2	bicycling
3	computer vision
4	gardening
5	gulf war
6	java
7	rock climbing
8	table tennis
9	vintage wine
10	volcano

In the second part of our evaluation, we compared the categorization of web pages by our method with those of Naive Bayes [88] and SVMLight [68].

All these methods including the proposed one were then used to obtain the topic categorization of ten randomly chosen pages (Table 4.4) into the fifteen topics under consideration, and the earlier volunteers were asked to rate them. A score of 10 to a page denotes that the volunteer viewed the page categorization as totally appropriate, while a score of 0 denotes a complete mismatch with the volunteer's categorization. The results are shown in Fig. 4.5 only for $\epsilon = 0.2$, as an example. Both the proposed algorithm and SVMLight produced significantly better categorization than the Naive Bayes algorithm at 95% confidence level. Since our algorithm has used the Naive Bayes algorithm for initial estimates, this indicates that our method has improved the categorization, as expected. However, it is seen that both ours and SVMLight are at par even at a 90% confidence level.

It may be noted that we had already obtained the joint probability matrix during our experiments on page ranking. Therefore, the categorization experiment only needed to implement the computations mentioned in Equations 4.28 and 4.25, which are very inexpensive, for each of the ten pages.

The number of iterations needed for the convergence of the proposed topic continuity model was 19 when ϵ was set to 0.35 and 14 when ϵ was chosen to be 0.2. For checking the scalability of the proposed algorithm, we measured the time taken by it and compared against the same for each of PageRank, QD-PageRank, and topic-sensitive PageRank (Table 4.5). The times mentioned are only those for the actual computation of the ranks and not for the preprocessing steps which are common to all. None of the above computations employed the extrapolation methods [70] mentioned above. It is worth mentioning that both the number of iterations and the time taken reported above do not include the corresponding figures for obtaining the intial estimates of the joint probability matrix.

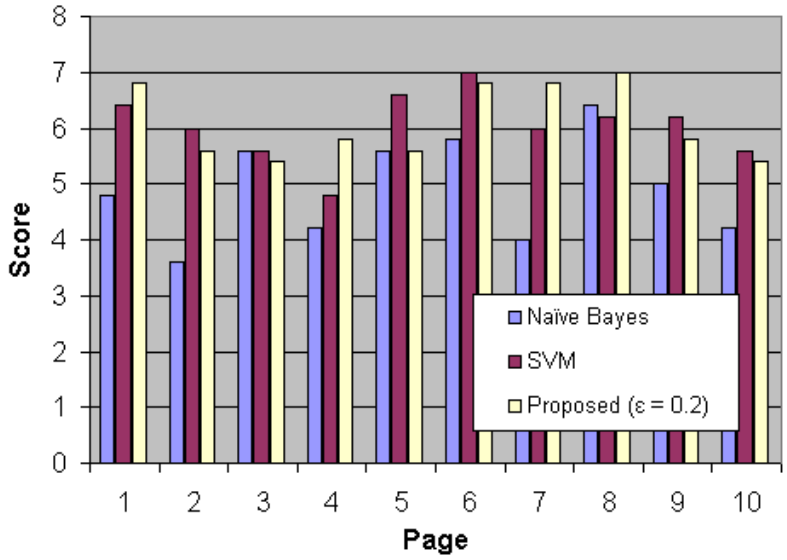


Figure 4.5: Comparison of categorization

4.5 Conclusions and Discussion

The problem of modeling the inter-relationship between page categorization and ranking in terms of topic continuity has been addressed in this chapter. An offline algorithm developed for this purpose probabilistically estimates the surfer's history, and thus, his/her current topic of interest. The incorporation of surfer history (or topic continuity) is a unique feature of this methodology. This resulted in a scalable and convergent iterative procedure that provides page categorizations as well as ranking simultaneously. The merits of the methodology have been established both theoretically and experimentally. Although we have presented experimental results only for page ranking and categorization, the method can be made applicable for topic-sensitive page ranking, topic representation on the web and personalization.

While the simultaneous estimation of page ranking and categorization is an advantage of the proposed method, a theoretical proof of convergence evades us for the same reason. Also, the topic continuity model presented in this chapter did not include the random jump factor. More recently, Nie *et al* [105], introduced another topic continuity model

Table 4.4: URLs of pages used for evaluating categorization

1	www.journalism.org/ccj/resources/symington.html
2	www.hcu.ox.ac.uk/TEI/P4beta/SA.htm
3	www.cruiseopinion.com/majesty-royal6.htm
4	www.icna.org/tm/feb00_cover5.htm
5	www.ashbrook.org/publicat/onprin/v8n4/hayward.html
6	www.osha.gov/oshstats/bls/txts/ostb0521.txt
7	www.pueblo.gsa.gov/cic_text/state/tips_canada.html
8	www.heritage.org/issues/chap5.html
9	www.skypub.com/news/news.shtml/spc/contact/news/news_print.html
10	www.state.mn.us/courts/library/archive/supct/9703/c9952124.htm

that incorporates the random jump factor, too.

In this chapter, we have dealt with web surfer models using probability theory where the performance depends on estimates of various parameters of the web graph, such as the probabilities of transition from one web page to another. Also, each surfer model makes some simplistic assumptions about the behavior of a hypothetical surfer. Acknowledging that the assumptions and estimates may be slightly off, one may like to have a methodology that serves a similar purpose as the aforementioned surfer models and, simultaneously, is fairly robust. Chapter 5 deals with one such attempt, namely, fuzzy web surfer models.

Table 4.5: Running time

Method	Time taken (in secs)
PageRank	60
QD-PageRank (for all 10 queries)	150
Topic-sensitive PageRank	1000
Proposed ($\epsilon = 0.35$)	1300
Proposed ($\epsilon = 0.2$)	1200

Chapter 5

Web Surfer Models Incorporating Fuzziness

5.1 Introduction

As the web consists of pages created by millions of individuals, there is a wide variety of authoring styles. Most present day content and link analysis algorithms are robust against differences in fonts, colors, *etc*., which are mostly ornamental. Some others can withstand, to some extent, malicious manipulation of content and links. However, they are sensitive to whether the information is contained in a single document or is spread out in a collection of documents. For the sake of uniformity in comparison during content and link analysis, information present in a single web page may be artificially divided into a collection of web pages. This division introduces an uncertainty in the page boundaries as well as the targets of hyperlinks.

A variety of web surfer models exist which model the sequence of web pages a surfer follows as a Markov process. The transition probabilities are obtained by considering the number of links in each page. Here, it is assumed that there is no uncertainty in the given web pages or the transition probabilities. In practice, this is not the case. This imprecision may be modeled with the help of fuzzy sets, or in particular, by fuzzy numbers. This

forms the basis for the present investigation, where we extend existing surfer models to fuzzy surfer models. Fuzzy web surfer models were first introduced in [104], and were studied in further detail in [15]. Since we now deal with fuzzy numbers, Markov chain theory is replaced by fuzzy Markov chain theory, which employs the max-min (or fuzzy) algebra instead of the classical algebra with multiplication and addition operations. These models may be employed, among other things, to compute ranks of web pages, which we call FuzzRanks. We believe that these models add to the set of tools needed for the development of intelligent information technologies [155] to be applied in the areas of web intelligence [92].

Fuzzy web surfer models described in this chapter, apart from being able to handle fuzziness in various aspects, inherit the advantages of fuzzy Markov models, namely, robustness and finite convergence. Robustness is a very important aspect because it implies that small changes in the transition matrix would not change the results drastically. Its significance arises from the fact that the transition matrices are not known beforehand and are estimated during the analysis phase, and so, (slightly) different methods of estimation, may lead to immensely dissimilar results. As a consequence, FuzzRank is more stable as compared to PageRank. FuzzRank reflects the belief of a surfer being on a page, and cannot fluctuate to extreme cases as in the case of probabilistic models.

The theory of fuzzy Markov chains is based on fuzzy algebra, also known as the $\max$-$\min$ algebra, which has been fairly well studied in literature, and well compared against classical algebra. Naturally, the question arises as to how models relying on these different algebras would compare against each other. After all, these models would inherit the advantages, as well as, the disadvantages of the underlying algebras, and it is worth a performance comparison. So, in addition to the lucrative properties of robustness and finite convergence, surfer models based on fuzzy Markov chains merit undertaking a study for purely academic reasons, too.

This chapter is organized as follows. Section 5.2 discusses the preliminaries such as fuzzy sets, Markov chains, fuzzy Markov chains and web surfer models. We make

use of these components to describe fuzzy web surfer models in Section 5.3, which we begin with a few motivational examples, and also define FuzzRank, which is the fuzzy-equivalent of PageRank. Section 5.4 consists of an illustrated example, and several experimental results, which convincingly demonstrate the advantages of FuzzRank over PageRank. Section 5.5, concludes the chapter and mentions some future directions of research on this topic.

5.2 Preliminaries and Background

We now provide the background as well as the notation on fuzzy sets, Markov chains, and fuzzy Markov chains.

5.2.1 Fuzzy Sets

Conventional sets consist of a group of elements. An element of the universe (Ω) may or may not belong to a given set, and only one of these two possibilities may happen. However, for the sake of situations where it is not clear if an element belongs to a set or not, the concept of fuzzy sets was proposed [152]. A fuzzy set is a generalization of the conventional set, where there is some measure of uncertainty of membership in the set. For a fuzzy set S, there is a membership function associated with it which provides a membership value for each element in Ω.

$$\mu_S : \Omega \rightarrow [0, 1] \tag{5.1}$$

Generally, μ is so chosen that $\max_{x \in \Omega} \mu_S(x) = 1$, in which case, it is said to be normalized.

The union and intersection operations of the classical sets is extended to the fuzzy sets, by taking the max and min, respectively, of the corresponding membership values of each element.

5.2.2 Markov Chains

A (first order) Markov chain [67] is a sequence $\{X_n\}_{n\in\mathbb{N}}$ of random variables where each random variable, X_i, takes a value from a state space S, and the sequence satisfies

$$P\left(X_{n+1}|X_0, X_1, \ldots, X_n\right) = P\left(X_{n+1}|X_n\right). \tag{5.2}$$

$\{X_n\}$ is called homogeneous if $P\left(X_{n+1}|X_n\right)$ is independent of n. In this chapter, we shall deal with only discrete, homogeneous Markov chains, with finite state space $S = \{1, 2, \ldots, N\}$. Let p_{ij} denote $P\left(X_{n+1} = i|X_n = j\right)$, which is the one step transition probability from state i to state j. $P = ((p_{ij}))_{i,j\in S}$ is called the (one-step) transition probability matrix. The probability of X_{n+1} assuming a state j is given by

$$\begin{aligned} P(X_{n+1} = j) &= \sum_{i=1}^{N} P(X_{n+1} = j|X_n = i)P(X_n = i) \\ &= \sum_{i=1}^{N} p_{ij}P(X_n = i) \end{aligned} \tag{5.3}$$

Now, the m-step transition probability from i to j, denoted by $p_{ij}^{(m)}$ may be expressed in terms of p_{ij} (which is the same as $p_{ij}^{(1)}$) as:

$$p_{ij}^{(m)} = P\left(X_{n+m} = i|X_n = j\right) \tag{5.4}$$

$$= \sum \prod_{k=1}^{m} P\left(X_{n+k} = x_{n+k}|X_{n+k-1} = x_{n+k-1}\right)$$

$$= \sum_{x_{n+1},\ldots,x_{n+m-1}\in S} \prod_{k=1}^{m} p_{x_{n+k},x_{n+k-1}} \tag{5.5}$$

From this expression, it may be observed that the m-step transition probability matrix $P^{(m)}$ is the same as P^m, the m-th power of P.

A state i is called aperiodic if $\gcd\{n : p_{ii}^{(n)} > 0\}$ is 1. A Markov chain is called aperiodic if all the states in S are aperiodic. It is called irreducible if every pair of states in S are reachable from each other. A finite, aperiodic, irreducible Markov chain is called regular, and $P^n > 0$ for some $n \geq 1$ for regular Markov chains.

For a regular Markov chain, $p_{ij}^{(n)} \rightarrow \pi_j \; \forall i, j \in S$. $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$ is called the stationary distribution of the Markov chain. This property is termed ergodicity, and means that, regardless of its initial state, $P(X_n = j)$ converges to a unique π_j. The convergence and uniqueness of the chain are guaranteed only if the chain is aperiodic and irreducible, respectively.

5.2.3 Fuzzy Markov Chains

The probabilities in the previous Subsection are real numbers and are all assumed to be known. In practice, they are estimated, and there are errors associated with the estimation procedure, which in turn, may again be estimated under suitable assumptions. The uncertainty in the transition probabilities may sometimes be better modeled in terms of fuzzy numbers.

In order to define a fuzzy Markov chain, we first define a fuzzy distribution and a fuzzy transition matrix.

A fuzzy distribution on S is defined by a mapping $\mu_x : S \rightarrow [0, 1]$, and is represented by a vector $\mathbf{x} = (\mu_x(1), \ldots, \mu_x(N))$.

A fuzzy transition matrix P is defined as a fuzzy distribution on the Cartesian product $S \times S$. P is represented by a matrix $((p_{ij}))_{i,j \in S}$ [4]. With this notation, a fuzzy Markov chain is defined as a sequence of random variables, where the transitions are determined by the fuzzy relation P and satisfy

$$\mu_{x^{(n+1)}}(j) = \max_{i \in S} \{\mu_{x^{(n)}}(i) \wedge p_{ij}\}, j \in S \tag{5.6}$$

Equation 5.6 is the fuzzy algebraic equivalent of the transition law of classical Markov chains provided in Eq. 5.3. The multiplication and addition operations in Eq. 5.3 have been replaced by the $\min$ and $\max$ operations, respectively. Naturally, the powers of the matrix P may be defined in the same manner as earlier. The interesting result is that, unlike the case of classical Markov chains, whenever the sequence of matrices P^n converges, it does so in finitely many steps to a matrix P^τ. If it does not converge, it

oscillates with a finite period ν starting from some finite power. The above statements are proved rather easily [4].

When the powers of P converge to a non-periodic solution P^τ, the associated fuzzy Markov chain is called aperiodic and P^τ is called a limiting fuzzy transition matrix. A fuzzy Markov chain is called ergodic if the rows of P^τ are identical. This definition is again similar to that of classical Markov chains, but the necessary and sufficient conditions for ergodicity are not known in this case [4].

We now propose a web surfer model that relies on the theory of fuzzy Markov chains. Since web surfer models and fuzzy Markov chains have been described in detail already, to avoid duplication, we describe the proposed methodology in a concise manner, making use of the notations and notions of this section.

5.3 Fuzzy Web Surfer

5.3.1 Motivation

We look at a few examples which demonstrate the need for new surfer models to deal with various kinds of uncertainty on the web. Authoring styles on the web vary widely and this results in the same kind of content being displayed in various formats. For example, the same content may be packed in one (possibly, big) document, or may be spread out across several linked list of documents. The process of retrieval and ranking are sensitive to such differences, which are usually a simple consequence of contrasting tastes or conveniences.

With most search engines indexing an increasing number of documents in PDF, PS and other formats, this situation is encountered all the more often. Fig. 5.1 shows a set of HTML pages, and a PDF document, both containing equivalent information. However, when the web is treated as consisting of individual atomic documents, it results in unfair comparison, as the PDF document has more content compared to each of the individual pages. One way to improve the level of fairness during such comparisons is to detect

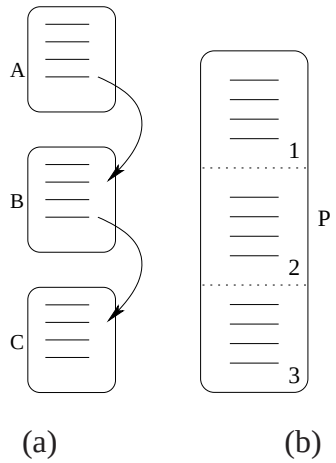


Figure 5.1: Equivalent information in (a) HTML and (b) PDF

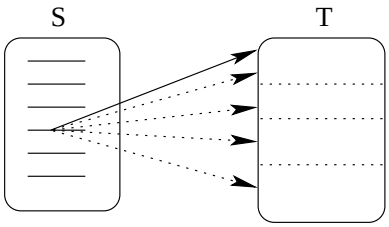
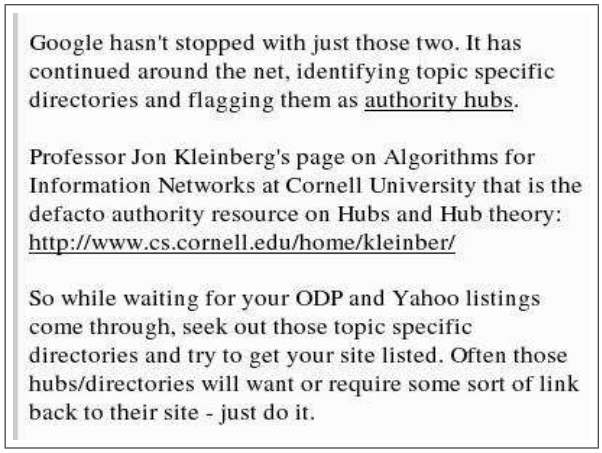


Figure 5.2: Which section is being pointed to? Actual target is fuzzy.

equivalent information, even if it is split across documents in one place and not the other, as performed in [102].

Fig. 5.2 shows a link from a source document S to a target document T. Now, T contains a lot of information, but the link is just for a particular portion of the page T. The question is which is the portion of T being implicitly referred to by this link. The relevance of this question lies in the fact that the link points to the web page T as a whole and so the weight being transferred through this link spills over to all of T instead of being restricted to the intended portion only.

To strengthen this argument, we provide a real life example. Fig. 5.3 shows a portion



Google hasn't stopped with just those two. It has continued around the net, identifying topic specific directories and flagging them as authority hubs.

Professor Jon Kleinberg's page on Algorithms for Information Networks at Cornell University that is the defacto authority resource on Hubs and Hub theory: http://www.cs.cornell.edu/home/kleinber/

So while waiting for your ODP and Yahoo listings come through, seek out those topic specific directories and try to get your site listed. Often those hubs/directories will want or require some sort of link back to their site - just do it.

Figure 5.3: A portion of a web page at Webmasterworld with a link to Jon Kleinberg's home page

of the web page located at `www.webmasterworld.com`[1] which contains a link to Jon Kleinberg's home page [2]. The link that leads to Kleinberg's home page provides no more information than its URL. The home page under consideration has two named sections, namely *Papers* and *Links*, and there is an introduction above it. It is clear from the context that the above mentioned link indeed refers to the *Papers* portion of the page. In addition, the *Papers* section is further subdivided according to the topics of the papers, but the subsections are not named. Had they been named, we can once again conclude that the link in question actually leads to the *Web Analysis and Search: Hubs and Authorities* subsection.

The above examples demonstrate that:

- a link to a web page may in reality be referring to just one or more pagelets, and not the whole page itself. Resolving which pagelet is referred to by a link needs contextual information, and yet this may not be precise.

- a page may have to be artifically divided into pagelets or sections, to avoid the weight attributed by a link to one pagelet spilling over to other pagelets. As men-

[1] `http://www.webmasterworld.com/forum10003/428.htm`

[2] `http://www.cs.cornell.edu/home/kleinber/index.html`

tioned earlier, this is required for fair comparison during retrieval because, although this particular link is for a small portion of the page, the contents of the rest of the page benefit from it, thus enjoying a better status as compared to similar content elsewhere.

It may be noted that it is not claimed that one or the other is necessarily better, because some systems may assign more weight to more content, whereas, others may penalize it. All that is being argued for is that such disparities may lead to diverse results, and need to be addressed at an early stage of link and content analysis. We now formulate a basic methodology for fuzzy web surfer models.

5.3.2 FuzzRank: Fuzzy Page Ranking

In what follows, we assume that the web pages have been preprocessed with the goal of increasing uniformity among them. By uniformity, we mean that the differences due to authoring styles, as explained earlier are reduced. There are two approaches for that. One approach is that big pages are split into pagelets [29, 120], and each of them can be called a new page. The other option is to merge related pages, making each of them a section (or pagelet) of one large page [101, 115]. The second approach is well suited for retrieval tasks by virtue of providing a larger coverage. However, for the purpose of link and content based analysis, we believe the first approach works better, because it generates a large number of small and coherent pages, thus avoiding topic drift.

As in existing surfer models, we label the available web pages (after preprocessing) from $\{1, 2, \ldots, N\}$. We propose the methodology for fuzzy web surfer models by imitating that of existing surfer models. Similar to the concept of PageRank, we define the concept of FuzzRank, where the objective is to compute, for each given web page, a value which reflects the belief that a web surfer would be on that page. This value is proportional to the belief that the surfer would be on one of its backlinks. Similarly, associated with each link in a page, there is a fuzzy number that indicates the belief that this

link would be followed, given that the surfer is on that page. These constitute the fuzzy transition matrix.

Formally, we are interested in computing $\mu(i)$ for each page i, which is the unconditional belief that a surfer would be on i. In other words, given a fuzzy transition matrix P, we want to obtain the eigen fuzzy set [4] of P which satisfies $\mu \circ P = \mu$. Here, the operation $\circ$ is the fuzzy max-min operation, as described in Eq. 5.6.

FuzzRank, the fuzzy counterpart of PageRank is now defined, as the greatest fuzzy eigen set of the fuzzy transition matrix, the existence of which has been proved in [129]. It is also known that this greatest fuzzy eigen set lies between $\mathbf{x^{(0)}}$ and $\mathbf{x^{(1)}}$, where

$$x_k^{(0)} = \min_j \max_i P_{ij} \; \forall k = 1, 2, \ldots, N$$

and

$$x_k^{(1)} = \max_i P_{ik} \; \forall k = 1, 2, \ldots, N.$$

$\mathbf{x^{(0)}}$ is always an eigen fuzzy set, whereas, whenever, $\mathbf{x^{(1)}}$ is an eigen fuzzy set, it is the greatest. Now, it is also known [4] that the greatest eigen fuzzy set is of the form of $\mathbf{x^{(1)}} \circ P^k$, for some positive integer k.

Thus, computing FuzzRank makes use of the power method in max-min algebra, and is similar to computing PageRank, the difference being that one cannot start with an arbitrary vector. One may note that, $\mathbf{x^{(1)}}$ itself equals $1 \circ P$, and hence FuzzRank is of the form $\mathbf{1} \circ P^k$. Thus the initial vector for the power iterations for computing FuzzRank is always $\mathbf{1}$.

So, the task at hand is to obtain accurate values of the elements of the fuzzy transition matrix, because, once that is done, Eq. 5.6 is all that is required to compute the FuzzRanks of the web pages.

Whenever a page has a single link to another page, it is assumed that there is no fuzziness present there. This is usually the case when an original page say A has been split into pagelets which were originally its named sections and a link from a page, say B had specifically pointed to a named section, say $A\sharp C$. Then, after splitting, the page

B points to just a single page representing $A \sharp C$. Had the link just pointed to A without referring to the intended section, the splitting would involve some fuzziness as to which section is being referred to. In that case, the membership values of the target of the link from B are non-zero for multiple pages representing the original sections of A. The membership values may be determined by considering similarity of the context around the anchor of the link and the potential target regions. Thus, the fuzzy transition matrix may be obtained.

5.3.3 Advantages and Limitations

We now discuss some features of the proposed class of fuzzy web surfer models. A list of advantages are listed first, following which we delve into the shortcomings of such a model.

We observe that theoretically, and intuitively, fuzzy web surfer models have the following merits:

1. Capture fuzziness in page contents: page boundaries may not be apparent all the time, especially when a single large page consists of several pagelets. Moreover, noise in web pages also affects the precise identification of the content of interest to the user.

2. Capture fuzziness in links: a page may contain several outlinks but not all of them may be intended for the same purpose. The reason for their presence may be ease of navigation, leading to advertisements, references, or pointing to authoritative resources. Similarly, a link to a particular page may in reality be actually for just one or two sections or pagelets of a page. These kinds of uncertainty may be better modeled by the proposed methodology.

3. Can take into account fuzzy contexts: context sensitive algorithms depend a lot on the modeling assumptions. For example, the context of a query may not be precisely clear, but the system may have a broad idea about it.

4. Robust computations: this is perhaps, the most emphatic reason for choosing fuzzy web surfer models. The computations in max-min algebra are more robust to perturbations as compared to usual addition and multiplication operations. There is an example in [4] that demonstrates the robustness of fuzzy Markov systems in comparison to regular Markov chains. When the entries of the transition matrix are perturbed by small quantities, the effects on the stationary distribution of the regular Markov chains are drastic, whereas, for fuzzy Markov chains, the changes are comparable to the perturbations.

5. Finite convergence: the stationary distribution of fuzzy Markov chains can be computed in finite number of steps, whereas, for regular Markov chains, only an approximation may be found as the convergence may not be achieved in finitely many steps. Existing web surfer models assume that, even though convergence is not attained, the order of the probabilities in the obtained distribution suffices.

We now study the possible limitations of the proposed methodology. It is well known that a Markov chain is ergodic if it is regular. However, in the case of fuzzy Markov chains, no such results are known. So, it is not clear when FuzzRank would actually exist, and even if it does, if it would be independent of the initial state of the process. There is an example in [4] where the rows of the limiting fuzzy transition matrix are distinct, thereby demonstrating the existence of non-ergodic fuzzy Markov chains.

This, however, need not be a limitation as all that it implies is that the final fuzzy distribution of the surfer being on a particular page may not be independent of his initial state. In practice, this may indeed be the case as a surfer starting from one set of pages may, in the long run, behave differently from another one who starts from a different set of web pages. Thus, that the fuzzy Markov chain of web pages being visited is not ergodic may be a blessing in disguise, which may be useful in computing topic sensitive page ranks or for detecting web communities.

5.4 Experimental Results

The objective of this section is to demonstrate the purpose and usefulness of fuzzy surfer models and to study the properties of FuzzRank. We begin this section with an example that serves as a preview of the experiments performed. The methodology is described as we present the details of the example.

The first step is to choose a (web) graph whose nodes are to be ranked. For this example, we choose a randomly generated directed graph. There are many generative graph models, as mentioned below:

- Erdos-Renyi model [47]: Given the number of vertices, edges are added randomly.
- Power-law models [97]: Here the in-degrees and out-degrees are assumed to arise from a power distribution of the form $y = x^a$. The R-MAT model [25] engulfs these and the Erdos-Renyi model.
- Lognormal models [18, 97]: These models have been shown to do better than the power-law models for modeling the web graph and are faster and scalable.

We choose the exponential distribution, which is similar to the lognormal distribution, and is easier to simulate. In our case, to make sure that there are no orphans (that is, nodes with no in-links), for each node, we draw a random number from a (zero truncated) exponential distribution, and choose that many nodes at random from which in-links to the present node are created. Note that the out-degree for some nodes may be zero. A sample random graph with 10 nodes is presented in Fig. 5.4.

The in-degrees, in-links, out-degrees and out-links for this sample graph are listed in Table 5.1. In this case, none of the out-degrees are zero.

Next, the PageRank vector is computed for this graph, and the vector at different iterations 1, 2, 3, and 31 (when it converged) are shown in Table 5.2. For the purpose of presenting consistently, all the vectors are $\max$-$\min$ normalized, that is, they are linearly transformed so that their minimum is 0 and maximum is 1. The value of d (the probability

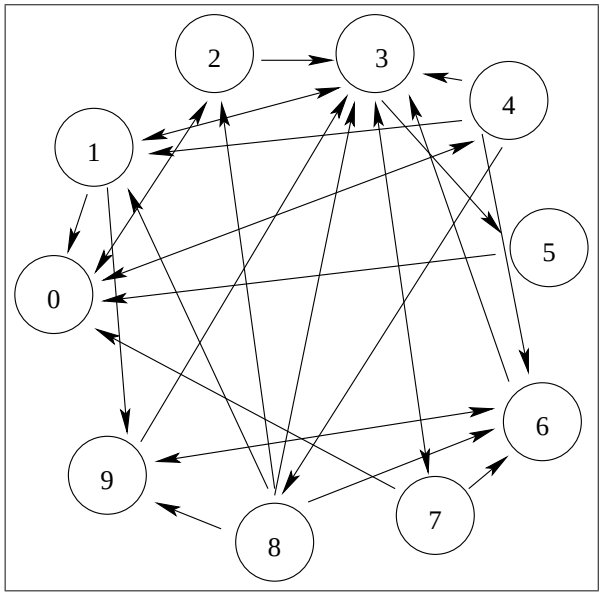


Figure 5.4: A sample graph with 10 nodes and 28 links

for a random jump) is set to 0.15 during the computation of PageRank throughout our experiments.

We provide the FuzzRank vectors, too, as they evolve over iterations, in Table 5.3. Throughout our experiments, we set the fuzzy transition matrix to be the same as that used by the random surfer model. Again, we note that, although no normalization is performed during the actual computation of FuzzRank, they are $\max$-$\min$ normalized while reporting them here.

We make the following observations by comparing Tables 5.2 and 5.3. We use Kendall discordance to measure the amount of disagreement between the two rank vectors. This discordance is defined as the proportion of discordant pairs among the total of $\frac{k(k-1)}{2}$ pairs. A pair (i, j) is called discordant with respect to two rank vectors, if i is ranked ahead of j by one, and ranked behind j by the other. In the case of a tie, it is assumed that there is no discernible disagreement.

- The ordering of pages according to FuzzRank is achieved after the first iteration itself, and the actual convergence requires one more iteration. For the case of PageR-

Table 5.1: In-Links and Out-Links of the Sample Graph in Fig. 5.4

Node	Out-Degree	Out-Links	In-Degree	In-Links
0	2	2, 4	5	1, 2, 4, 5, 7
1	3	0, 3, 9	3	3, 4, 8
2	2	0, 3	2	0, 8
3	3	1, 5, 7	7	1, 2, 4, 6, 7, 8, 9
4	5	0, 1, 3, 6, 8	1	0
5	1	0	1	3
6	2	3, 9	4	4, 7, 8, 9
7	3	0, 3, 6	1	3
8	5	1, 2, 3, 6, 9	1	4
9	2	3, 6	3	1, 6, 8

Table 5.2: PageRank computations for the Sample Graph in Fig. 5.4

Node	Initial	Iter 1	Iter 2	Iter 3	...	Iter 31
0	1.000	0.915	0.618	0.935	...	0.879
1	1.000	0.225	0.430	0.326	...	0.365
2	1.000	0.211	0.527	0.225	...	0.390
3	1.000	1.000	1.000	1.000	...	1.000
4	1.000	0.126	0.494	0.201	...	0.359
5	1.000	0.056	0.339	0.200	...	0.242
6	1.000	0.436	0.370	0.374	...	0.367
7	1.000	0.056	0.339	0.200	...	0.242
8	1.000	0.000	0.000	0.000	...	0.000
9	1.000	0.352	0.412	0.279	...	0.327

Table 5.3: FuzzRank computations for the Sample Graph in Fig. 5.4

Node	Initial	Iter 1	Iter 2	Iter 3
0	1.000	1.000	1.000	1.000
1	1.000	0.166	0.444	0.444
2	1.000	0.375	1.000	1.000
3	1.000	0.375	1.000	1.000
4	1.000	0.375	1.000	1.000
5	1.000	0.166	0.444	0.444
6	1.000	0.375	1.000	1.000
7	1.000	0.166	0.444	0.444
8	1.000	0.000	0.000	0.000
9	1.000	0.375	1.000	1.000

ank, the final ordering is achieved after the twelfth iteration.

- FuzzRank has clumped several nodes together. The number of distinct ranks are 3 for FuzzRank and 9 for PageRank. This indicates that FuzzRank is a more conservative way of ranking (compared to PageRank) where it concludes that the given information in the form of the structure of the graph is insufficient for strictly putting one node ahead of the other, and encourages the use of other factors, such as query relevance, to make this decision.

- The discordance between PageRank and FuzzRank is $\frac{2}{45}$. The disagreement is due to a single node (Node 1), which has a PageRank of 0.365. Had the PageRank value been 0.326 or less (with the PageRank values remaining the same for the remaining nodes), there would have been no discordance between the two rank vectors.

Often, one would be interested in finding the discordance between the top k ranked nodes. Generally, this is to indicate that a discordant pair among the top ranked pages

of a ranked list is more significant as compared to the same at the bottom of the list. Comparing top k lists [48] involves obtaining the top k elements of both the lists and looking for discordant pairs among the union of those elements. If c is the number of elements in common to the two top k lists, the total number of elements in the union, n, is $k-c+k-c+c = 2k-c$. Unlike in [48], where it is assumed that the ranks of the nodes outside the top k lists are not known, we shall make use of the available information to compute the actual discordance, thereby avoiding the estimation of discordance suggested in [48]. To handle ties consistently, we shall keep all the tied elements together. So, for the FuzzRank vector, the sets top 1 to top 6 are all the same, consisting of the nodes $0, 2, 3, 4, 6,$ and 9, whereas the top 7 to top 9 lists have the nodes $1, 5,$ and 7, in addition to the aforementioned 6 nodes. As we vary k from 1 (we note that, for $k = 1$, the top elements of both lists may be the same, and hence a pair might not be possible at all, in which case the discordance would be defined to be zero) to 10, the number of discordant pairs between PageRank and FuzzRank is 0 for $1 \leq k \leq 5$, 1 for $k = 6$, and 2 for $7 \leq k \leq 10$. Fig. 5.5 plots the normalized discordance values, whereby, for each $k > 1$, the number of discordant pairs is divided by $\frac{n(n-1)}{2}$.

We now look at the effects of mutating the given graph on the rankings of the nodes. This is important because, often, a link to a page (or a section of the page), might not exist explicitly. The ranking algorithm would need to be robust to gracefully handle noise in links. It has been shown theoretically in [4] that the classical Markov chains may be severely impacted by small changes in the transition matrix, whereas, that is not the case for fuzzy Markov chains.

We perform a simple mutation on the sample graph, such as adding a node or removing a node. Each of the following three graphs is produced as a result of one such mutation.

M1: The link $4 \rightarrow 3$ is removed.

M2: The link $5 \rightarrow 0$ is removed. Note that, this means that there are no out-links from node 5, making it a *rank leak*.

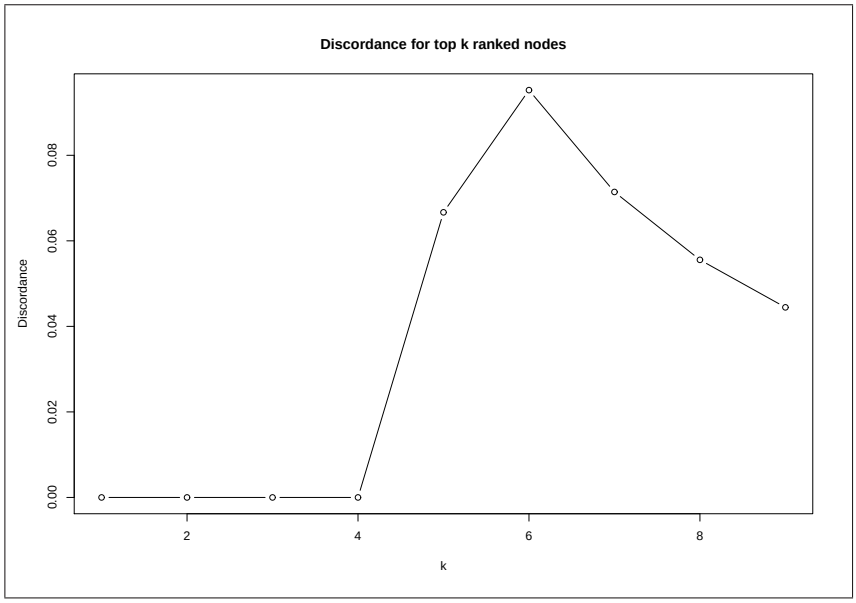


Figure 5.5: Discordance between the PageRank and FuzzRank vectors for the top k ranked nodes of the sample graph

M3: A new link $5 \rightarrow 2$ is added to the graph.

The PageRank and FuzzRank vectors are computed for each of the mutated cases, and are presented alongside those for the original graph. Note that, there is no discordance between the FuzzRank vectors for the various graphs, although, the scores have changed for some of the nodes. For the case of PageRank, however, each mutation produces a different change. This demonstrates the robustness of FuzzRank. The significance of this robust computation is that the creation of the transition matrix is based on several (simplistic) assumptions, and when these deviate from reality, the resultant ranking may be well away from the ideal one. The robustness of FuzzRank is related to the great number of ties in this case. By not committing itself to a strict ranking, it absorbs the effects of slight changes in the transition matrix.

We now describe the data sets that we have used in our experiments. We have generated 100 graphs each of sizes 10, 100, and 1000, randomly. We chose two real life data sets from Stanford's WebBase [65] and named them WB1_7440 and WB4_7060, after the host and port numbers from which they are available. The former is a crawl of a part of

Table 5.4: PageRank and FuzzRank for the nodes in the Sample Graph in Fig. 5.4, and its mutated versions, with the nodes ordered in descending order of PageRank

Node	PageRank	FuzzRank	PR M1	FR M1	PR M2	FR M2	PR M3	FR M3
3	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
0	0.879	1.000	0.950	1.000	0.475	1.000	0.726	1.000
2	0.390	1.000	0.412	1.000	0.256	1.000	0.505	1.000
6	0.367	1.000	0.412	1.000	0.403	1.000	0.349	1.000
1	0.365	0.444	0.386	0.333	0.369	0.444	0.358	0.444
4	0.359	1.000	0.373	1.000	0.213	1.000	0.300	1.000
9	0.327	1.000	0.356	1.000	0.374	1.000	0.316	1.000
5	0.242	0.444	0.220	0.333	0.268	0.444	0.251	0.444
7	0.242	0.444	0.220	0.333	0.268	0.444	0.251	0.444
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

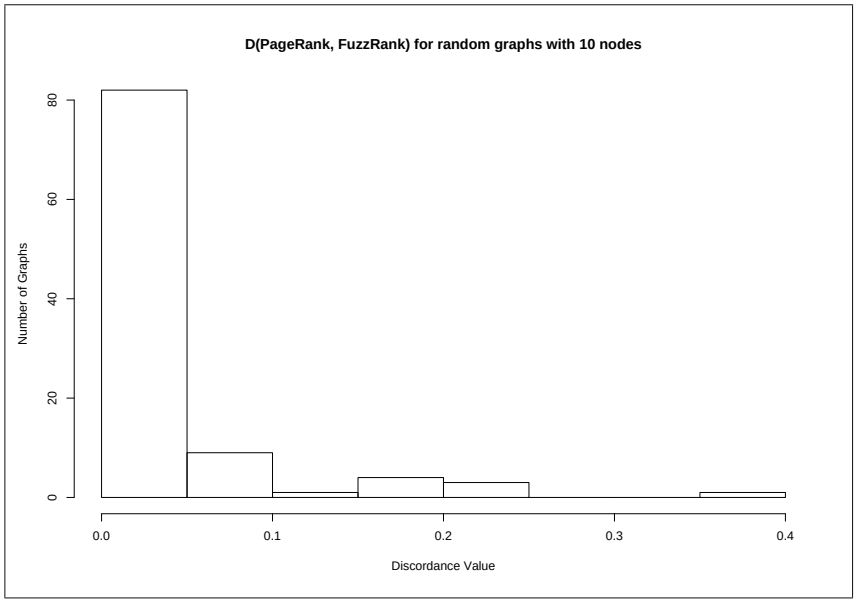


Figure 5.6: Discordance values between PageRank and FuzzRank vectors for 100 randomly generated graphs with 10 nodes

the berkeley.edu domain, and there are about 140 thousand pages with over 1.6 million links to pages within the same data set. WB4_7060, which is a crawl of a part of the stanford.edu domain, consists of about 40 thousand pages and over 260 thousand links to pages within itself.

Having earlier detailed the methodology of the experiments on an example, the results are now quickly presented. Figs. 5.6, 5.7, and 5.8 present the discordance values between the PageRank and FuzzRank vectors for 100 randomly generated graphs with 10, 100, and 1000 nodes, respectively. As in the earlier example, the discordance values are generally low. Moreover, these values decrease as the number of nodes increases. The corresponding discordance values for the WB1_7440 and WB4_7060 data sets are 0.08 and 0.1, respectively, implying that PageRank and FuzzRank do not disagree much on even large real life data sets.

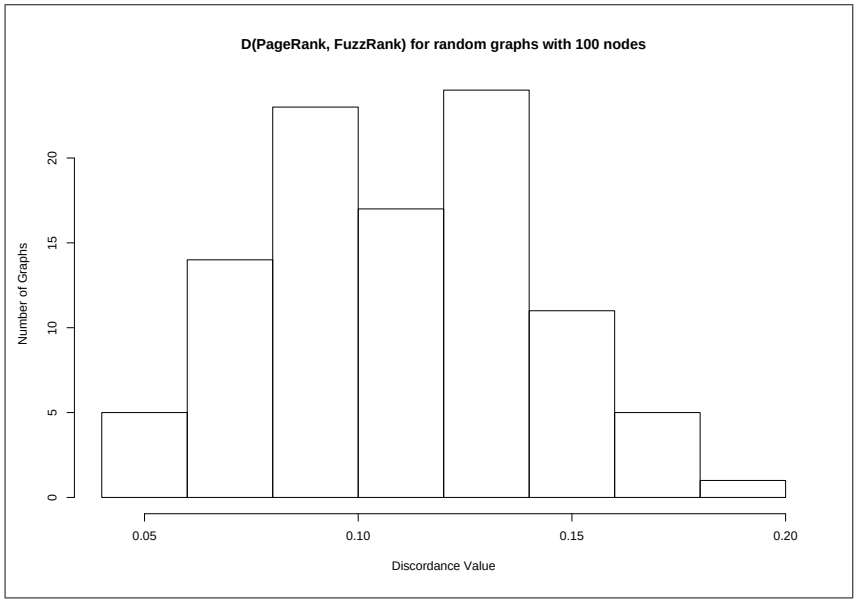


Figure 5.7: Discordance values between PageRank and FuzzRank vectors for 100 randomly generated graphs with 100 nodes

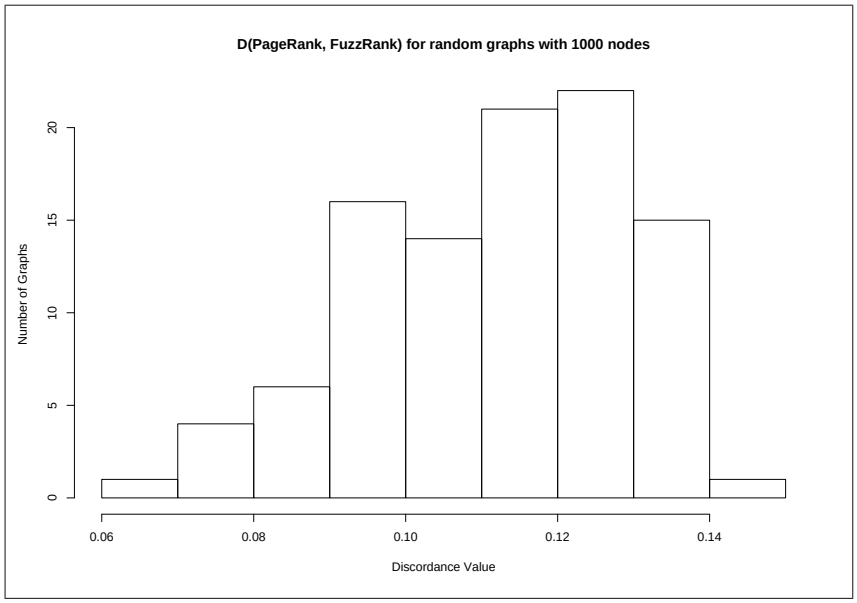


Figure 5.8: Discordance values between PageRank and FuzzRank vectors for 100 randomly generated graphs with 1000 nodes

5.5 Conclusions and Discussion

The novel theoretical formulation of fuzzy web surfer models by integrating existing works on web surfer models and fuzzy Markov chains is very interesting. The definition of FuzzRank as the fuzzy surfer models counterpart of PageRank, which is based on the random surfer model is both simple and elegant. Experimental results confirm that FuzzRank has very similar ranking properties, and yet is more robust to noise. This robustness is a consequence of FuzzRank to avoid a strict ranking in the absence of strong evidence to that effect. While this may result in a large number of ties if this were the sole criterion for ranking web pages, given that several other factors, like query relevance, would be considered during the ranking process, the ability to consistently rank the pages in the presence of noise is an advantage. The walk-through with an example clearly shows how stable FuzzRank is over PageRank. Future directions for research on this topic involve stability analysis for various other kinds of noise, and obtaining an accurate fuzzy transition matrix based on both the links and the contextual information.

The present chapter, along with the previous one, dealt with ranking web pages. In the next chapter, we discuss how to compare page rank vectors, or more generally, any pair of ranking schemes.

Chapter 6

Quantitative Evaluation of Page Ranking Schemes

6.1 Introduction

Ranking a set of items is a fairly frequent task, and involves pairwise comparison of the given items. This comparison may be performed by inquiring an oracle for each pair of items, in which case, the ranking procedure is known as comparison based ranking. On the other hand, one may assign scores to each item, thus producing a total ordering on the set of items. Each item is assigned a score which denotes how early the item appears in the list, and thus, comparing each pair is now performed by comparing the corresponding scores. The present work is concerned with score based rankings only, and each ranking is assumed to be induced by a *scoring scheme* or *function*.

Several scoring schemes may compete with each other for ranking the same set of items, and the items may be ranked differently by each of them. Given two such scoring schemes, two questions arise:

- Which scheme is better?
- How different are the two schemes?

In the present chapter, we are concerned with the answers to the second question. One may note that it is not sufficient to ask the first question alone as, instead of just declaring one of them to be better than the other, it is imperative to measure how much better one scoring is over the other.

Comparing such scoring schemes is generally performed by comparing the induced ranking on the set of items. However, several different scoring systems may lead to the same ranking of the items, and a rank based comparison cannot discriminate between such schemes. In the present chapter, we propose a more general approach, whereby, the scoring schemes may be perceived to be different even if they induce identical rankings.

Our approach is based on the idea that similar scoring schemes discriminate between two items in a similar manner. If the scores assigned to items i and j by one scoring scheme are far apart, while those by another are very close to each other, it indicates that the two schemes are dissimilar. It is interesting to note that this also corresponds to a fusion based approach for measuring similarity/dissimilarity of scoring schemes. Often, these scoring schemes are used in combination with some other scoring method, say T, to produce the final ranking [34, 44, 108, 122]. This process of combining scores is referred to as *score fusion* [44, 108]. If two sets of scores are exactly the same, their rankings remain the same even after score fusion. Also, differences in the scores assigned by two methods may lead to different rankings, depending on the scores used for fusion. If we know T beforehand, then we may rank the items after fusing their scores, and compute a dissimilarity value on the basis of the induced rankings.

In the present chapter, we look at the case where T remains unknown. Based on certain simple assumptions about this unknown scoring scheme T, we compare two scoring schemes on the basis of how likely they are to produce a discordant pair. We provide a metric in this regard, which relies on the margins separating the scores. Even if two items receive almost equal scores, they might be ranked differently depending on T. The present investigation is about studying how likely it is for them to be ranked differently upon score fusion.

The manuscript is organized as follows. We introduce the notation and background for comparing scores and rankings, and rank fusion in Section 6.2. The proposed methodology is described in Section 6.3, which includes motivational examples and a discussion on the characteristics of our method. Section 6.4 deals with extending the proposed metric for comparing top k scorings, and applications of the metric are discussed in Section 6.5. We report some preliminary experimental results in Section 6.6, before drawing our conclusions and mentioning future work in Section 6.7.

6.2 Comparing Scoring Functions: Background

6.2.1 Notation

Our universe consists of a set of objects or items indexed by $\Omega = \{1, 2, 3, \ldots, n\}$, and each of them shall be referred to by its index. Unless otherwise stated, i and j refer to two elements of Ω, and $i < j$. These objects or items may be documents in corpus, states in a country, students in a university, and so on. A scoring scheme or function assigns a real number $s_i \in [0, 1]$, called a score, to i, for each $i \in \Omega$. In the present work, only normalized scoring schemes shall be considered, i.e., $\max_i s_i = 1$ and $\min_i s_i = 0$. The k^{th} scoring scheme, or equivalently, the k^{th} score vector is denoted by $S_k = (s_{k1}, s_{k2}, \ldots, s_{kn})$. We use $R(s_i)$ to denote the rank of i, and $R(S)$ is an abbreviation for $(R(s_1), R(s_2), \ldots, R(s_n))$. Objects with larger scores appear earlier in the ranked list, so that $s_i > s_j \Rightarrow R(s_i) < R(s_j)$.

Let S_1 and S_2 be two scoring schemes. A pair (i, j) is called discordant w.r.t. S_1 and S_2, if $(s_{1i} - s_{1j})(s_{2i} - s_{2j}) < 0$, i.e., the two schemes order i and j in different ways. If $(s_{1i} - s_{1j})(s_{2i} - s_{2j}) > 0$, then i and j are said to be concordant. The comparison is called a tie if $(s_{1i} - s_{1j})(s_{2i} - s_{2j}) = 0$. A tie may occur in one of three ways: $s_{1i} = s_{1j}$, $s_{2i} = s_{2j}$, or both, and it is called a 1-tie, a 2-tie or a double tie, respectively. Without loss of generality, we assume that the first set of scores are sorted:

Assumption 1 (Monotonicity of S_1) $1 = s_{11} \geq s_{12} \geq \ldots \geq s_{1n} = 0,$

for, otherwise, we may sort S_1 and S_2 in descending order with S_1 as the primary key.

Let S and $T = (t_1, t_2, \ldots, t_n)$ be two score vectors. $\alpha S + \beta T$ means the score vectors S and T are *fused together*, with the fusing proportions $0 < \alpha < 1$ and $\beta = 1 - \alpha$. So, the i^{th} element of the fused vector is $\alpha s_i + \beta t_i$. When $\alpha = \beta = 0.5$, we shall simply write $S + T$, instead of the technically correct $0.5S + 0.5T$, noting that the ranking remains the same in both cases.

We now provide some background on comparing rankings and fusion.

6.2.2 Background on Rank Comparison

Comparison of rankings is a fairly well studied problem, and we mention the most popular rank comparison methods here. Comparing two different rankings has been studied in various fields. In each case, a measure has been provided that takes into account how much the positions of each item differ in the two ordered lists. The measure is zero when the two rankings are exactly the same, whereas it is maximum when the rankings are completely opposite to each other. Some very useful and widely used measures for comparing two rankings are Spearman's footrule, Spearman's rank correlation and Kendall's τ.

Spearman's footrule is defined as:

$$\rho_1 := \sum_{i=1}^{n} \left| R(s_{1i}) - R(s_{2i}) \right|. \tag{6.1}$$

Spearman's rank correlation [32] is defined as:

$$\rho_2 := \left(\sum_{i=1}^{n} \left(R(s_{1i}) - R(s_{2i}) \right)^2 \right)^{\frac{1}{2}}. \tag{6.2}$$

Both ρ_1 and ρ_2 are 0 if both the rankings are the same, and attain their maximum values when $R(S_{1i}) = n + 1 - R(S_{2i}), \forall\, i \in \Omega$.

Kendall's Tau (or τ) [32] is defined as the difference of the proportions of concordant

and discordant pairs according to S_1 and S_2:

$$\tau(S_1, S_2) = \frac{2}{n(n-1)} \sum_{i<j} sign(s_{1i} - s_{1j})(s_{2i} - s_{2j}) \tag{6.3}$$

and may be rewritten in terms of only the number of discordant pairs as

$$\tau(S_1, S_2) = 1 - \frac{4}{n(n-1)} \sum_{i<j} I_{[(s_{1i}-s_{1j})(s_{2i}-s_{2j})<0]} \tag{6.4}$$

The summation in Eq. (6.4) is the number of discordant pairs w.r.t. S_1 and S_2 and is referred to as the Kendall distance between them [46]. Formally, the Kendall distance between each pair $\{i, j\}$ w.r.t. S_1 and S_2 is defined as:

$$K(S_1, S_2; i, j) = \begin{cases} 1 & \text{if} \quad i \text{ and } j \text{ are discordant} \\ \frac{1}{2} & \text{if} \quad i \text{ and } j \text{ have a single tie} \\ 0 & \text{o.w.,} \quad \text{i.e., if } i \text{ and } j \text{ are concordant or have a double tie} \end{cases} \tag{6.5}$$

When this quantity is summed over all the pairs of items, then it is called the Kendall distance between S_1 and S_2, or equivalently, between $R(S_1)$ and $R(S_2)$, and is denoted by $K(S_1, S_2)$ or $D(R(S_1), R(S_2))$. This is also referred to as Kendall (Tau) distance, Kemeny distance, or bubblesort distance between $R(S_1)$ and $R(S_2)$ when interpreted as the number of pairwise adjacent transpositions needed to transform from one ranked list to the other.

More recently, Bar-Ilan, et al. proposed that differences in ranking in the initial part of the lists should be given more weightage than those towards the end of the lists [9]. The dissimilarity between the two rankings is computed as:

$$\mu = \sum_{i=1}^{n} \left| \frac{1}{R(S_1(i))} - \frac{1}{R(S_2(i))} \right| \tag{6.6}$$

6.2.3 Background on Fusion

Fusion is the process of combining multiple sets of ranks or scores available for the given items. Rank fusion [100, 130], also known as rank aggregation [46, 151], obtains a con-

sensus ranking from the available ranked lists. These lists need not be full lists, making rank fusion a very challenging problem.

Score fusion, on the other hand, combines the scores directly, in order to produce a consensus score vector, on which the final ranking may be based upon. Such fusion may be performed by taking an average of the scores assigned to an item. Two of the standard score fusion techniques are CombSUM (a simple average) and CombMNZ (a weighted average) [83, 122, 143].

Several studies have compared the effectiveness of rank and score fusion. Scores contain more information than ranks, but may be prone to noise. It is suggested in [46] that only the induced ranks should be considered for fusion, whereas, in [99], it is found that score fusion is advantageous, provided that normalization is performed properly. A detailed discussion on ranks versus scores is available in [98].

6.3 Comparing Underlying Scores Directly

One may compare two score vectors directly using a measure like Pearson's correlation coefficient. However, the interpretation of the coefficient in terms of the resultant ranking is lost. Also, the correlation coefficient is not a metric, and hence cannot be interpreted directly as a distance between two scoring systems. The correlation coefficient may be transformed into a metric, but it still does not reflect the rank-specific differences between two scorings, and is not always useful for comparing rank-inducing scoring functions.

An alternative is to compare the scores assigned to the available objects on the basis of the rankings they produce.

6.3.1 Motivation

We shall now emphasize upon the significance of comparing scorings directly. We start by asking the following question: "Is it sufficient to use only $R(S_1)$ and $R(S_2)$ for comparing S_1 and S_2?" We look at the following examples to gain some insight in this regard.

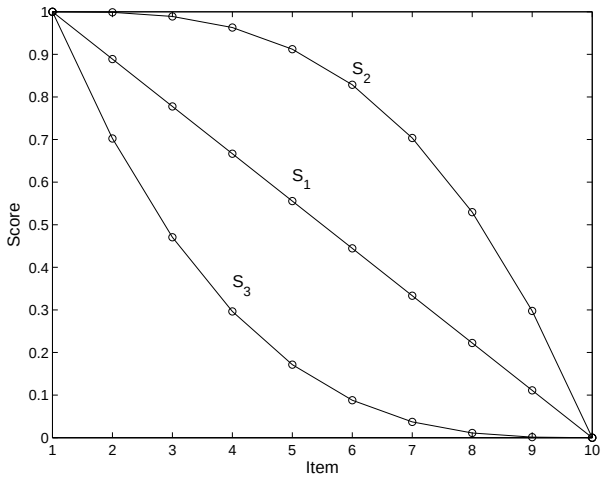


Figure 6.1: Same ranks, different scores

Example 1: Suppose that items i and j receive identical scores in each scoring scheme, i.e., $s_{1i} = s_{1j}$ and $s_{2i} = s_{2j}$. By definition, there is a double tie between i and j w.r.t. S_1 and S_2. Now, if there is a measurement error, due to which the scores are slightly perturbed, i and j would be declared to be either concordant or discordant (with probability 1), though, in reality, they are neither of the two. This is a consequence of the fact that discordance is a hard concept, and a pair may be either discordant or not, but nothing in between.

Example 2: Let $n = 10$. The objects $1, 2, \ldots, 10$ are scored in three different ways as shown in Fig. 6.1. Here, $s_{1i} = \frac{n-i}{n-1}$, $s_{2i} = 1 - \frac{(i-1)^2}{(n-1)^2}$, $s_{3i} = \frac{(n-i)^2}{(n-1)^2}$. It may be noted that, while the ranks are identical in all three cases, the scores differ significantly. For example, items 1, 2 and 3 have barely distinguishable scores w.r.t. S_2, whereas, the scores are quite varied in the cases of S_1 and S_3. If ranking by themselves is the sole objective of the three scoring functions, then they may be deemed identical. Otherwise, that the resolving power of the three scoring functions is different implies some amount of dissimilarity between them.

Example 3: Let $(s_{1i}, s_{1j}, s_{2i}, s_{2j}, s_{3i}, s_{3j}) = (0.4, 0.5, 0.5, 0.4, 0.9, 0.1)$. So, the items i and j are discordant w.r.t. S_1 and S_2, as well as, w.r.t. S_1 and S_3. However, are they

"more" discordant in the second case? Again, this question cannot be answered without the notion of a degree of discordance.

Example 4: Let $s_{12} = 0.7, s_{13} = 0.4, s_{22} = 0.6$ and $s_{23} = 0.5$. Assume $s_{1k} = s_{2k}\ \forall k \in \Omega \setminus \{2, 3\}$. Note that items 2 and 3 are concordant w.r.t. S_1 and S_2, and also that both S_1 and S_2 induce the same rankings, i.e., $R(S_1) = R(S_2)$. Now, suppose that the objects are to be ranked after fusing their scores with T. So, the two rankings obtained are $R(S_1 + T)$ and $R(S_2 + T)$. The question we are concerned with is whether these two rankings are identical. It is obvious that the answer depends on the values of t_2 and t_3. For example, if $t_2 = 0.3$ and $t_3 = 0.5$, then the fused scores are given by $s_{12} + t_2 = 1.0 > s_{13} + t_3 = 0.9$ and $s_{22} + t_2 = 0.9 < s_{23} + t_3 = 1.0$, and hence $(2, 3)$ forms a discordant pair according to $S_1 + T$ and $S_2 + T$. One may easily observe that $(2, 3)$ would form a discordant pair whenever $t_3 - t_2 \in (0.1, 0.3)$.

Example 5: Now, if $s_{12} = 0.9$, and $s_{13} = 0.1$, while s_{22} and s_{23} remain the same as in Example 4, 2 and 3 again form a concordant pair w.r.t. S_1 and S_2. However, $(2, 3)$ forms a discordant pair w.r.t. $S_1 + T$ and $S_2 + T$ whenever $t_3 - t_2 \in (0.1, 0.8)$. In a sense, it is more likely for $(2, 3)$ to be discordant in this case, than in the earlier one.

The essence of these examples was to demonstrate that even though S_1 and S_2 may appear identical or similar on the basis of the rankings they produce by themselves, the likelihood of a discordant pair being produced after score fusion depends both on the distribution of $t_j - t_i$ (for all pairs $i < j$) and the spacing between the scores assigned to the objects of the universe.

Another compelling reason for comparing scores directly is that given just S_1 and S_2, rank comparison methods have no way of distinguishing between the cases when the fusing parameter, α, is big (say, 0.9) or small (say, 0.1). In isolation, as long as both the vectors are multiplied by the same scalar, rank comparison measures come up with the same value each time. Of course, if all one needs to do is to rank the items on the basis of just S_1 and S_2, then $R(S_1)$ and $R(S_2)$ should suffice for comparing the scores, i.e., score comparison methods provide no additional advantage.

6.3.2 Comparing Scores Directly

The objective in the present investigation is to discern between two scoring functions directly without performing the additional task of computing the induced ranks. Taking a cue from Examples 1 and 3, we propose the concept of a degree of discordance for a pair of items, which subsumes the usual definition of discordance as a special case. As discussed in Example 2, the dissimilarity of two scoring functions w.r.t. a pair $\{i, j\}$ may be inferred from the differences in the separation of i and j by the scoring functions. Thus, a measure of dissimilarity between S_1 and S_2 may be based on the separations $d_{ij}^{(1)} = s_{1i} - s_{1j}$ and $d_{ij}^{(2)} = s_{2i} - s_{2j}$. The more $d_{ij}^{(1)}$ and $d_{ij}^{(2)}$ are apart, the higher the dissimilarity.

We now look at an alternative approach, which leads to the same notion of discordance once again. In particular, we would like to study how likely it is for a discordant pair to appear during score fusion.

In this regard, let us formalize our observations in Examples 4 and 5 of Section 6.3.1. The fused scores of i w.r.t. $S_1 + T$ and $S_2 + T$ are $s_{1i} + t_i$ and $s_{2i} + t_i$, respectively, for each $1 \leq i \leq n$. The pair (i, j) forms a discordant pair w.r.t. $S_1 + T$ and $S_2 + T$, if and only if,

$$((s_{1i} + t_i) - (s_{1j} + t_j))\,((s_{2i} + t_i) - (s_{2j} + t_j)) < 0,$$

or equivalently, if and only if,

$$((t_j - t_i) - (s_{1i} - s_{1j}))\,((t_j - t_i) - (s_{2i} - s_{2j})) < 0. \tag{6.7}$$

That the quantity (6.7) is negative is equivalent to having $(t_j - t_i)$ in the interval

$$\left(\min\left\{d_{ij}^{(1)}, d_{ij}^{(2)}\right\}, \max\left\{d_{ij}^{(1)}, d_{ij}^{(2)}\right\}\right),$$

where, $d_{ij}^{(1)}$ and $d_{ij}^{(2)}$ denote the differences $s_{1i} - s_{1j}$ and $s_{2i} - s_{2j}$, respectively. Thus, once again, the dissimilarity is proportional to the difference of $d_{ij}^{(1)}$ and $d_{ij}^{(2)}$. Note that $d_{ij}^{(1)}$ is positive, by Assumption 1, in Section 6.2.1.

Similarly, it may be easily seen that the pair (i, j) forms a discordant pair w.r.t. $\alpha S_1 + \beta T$ and $\alpha S_2 + \beta T$, if and only if $t_j - t_i$ belongs to the interval

$$\left(\min \left\{ \frac{\alpha}{\beta} d_{ij}^{(1)}, \frac{\alpha}{\beta} d_{ij}^{(2)} \right\}, \max \left\{ \frac{\alpha}{\beta} d_{ij}^{(1)}, \frac{\alpha}{\beta} d_{ij}^{(2)} \right\} \right).$$

Let γ denote the ratio $\frac{\alpha}{\beta}$, and let, for each pair $i < j$,

$$\left.\begin{array}{rcl} a_{ij}^{S_1,S_2} & = & \min \left\{ s_{1i} - s_{1j}, s_{2i} - s_{2j} \right\}, \text{and} \\ b_{ij}^{S_1,S_2} & = & \max \left\{ s_{1i} - s_{1j}, s_{2i} - s_{2j} \right\}. \end{array}\right\} \tag{6.8}$$

We note that, for each pair (i, j), there are associated real numbers $a_{ij}^{S_1,S_2}$ and $b_{ij}^{S_1,S_2}$, such that (i, j) is a discordant pair according to $\alpha S_1 + \beta T$ and $\alpha S_2 + \beta T$ whenever $t_j - t_i$ is in the interval $\left(\gamma a_{ij}^{S_1,S_2}, \gamma b_{ij}^{S_1,S_2} \right)$. For ease of notation, we drop the superscripts S_1 and S_2 when they are clear from the context. As seen earlier, the interval $[a_{ij}, b_{ij}]$ holds the key to the likelihood of a discordant pair being produced. For this reason, we propose

$$D_\gamma(S_1, S_2) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} D_\gamma(S_1, S_2; i, j), \tag{6.9}$$

as a measure of discordance between S_1 and S_2, where $D_\gamma(S_1, S_2; i, j)$ is a suitably chosen measure on the interval (a_{ij}, b_{ij}). We shall write D as a shorthand for D_1. In the present chapter, we make the choice of $D_\gamma(S_1, S_2; i, j)$ as

$$D_\gamma(S_1, S_2; i, j) = \int_{\gamma a_{ij}}^{\gamma b_{ij}} f(x)dx = \left| \int_{\gamma(s_{1i} - s_{1j})}^{\gamma(s_{2i} - s_{2j})} f(x)dx \right|, \tag{6.10}$$

where, $f(x)$ is a continuous probability density function with $(-1, 1)$ as its support. As a particular case, in the present work, we choose f to be the triangular density function:

$$f(x) = \begin{cases} 1 + x & \text{if} \quad -1 < x \leq 0 \\ 1 - x & \text{if} \quad 0 < x \leq 1 \\ 0 & \text{o.w.} \end{cases} \tag{6.11}$$

$D_\gamma(S_1, S_2; i, j)$ ranges between 0 and 1 and shall be called the degree of discordance between i and j w.r.t. S_1 and S_2. A value of 0 denotes either a double tie or perfect

concordance (i.e., $d_{ij}^{(1)} = d_{ij}^{(2)}$), whereas 1 implies perfect (or extreme) discordance (i.e, $d_{ij}^{(1)} d_{ij}^{(2)} = -1$).

The significance of choosing this particular function f is as explained below. The relevance scores lie in the interval $[0, 1]$, and thereby, $t_j - t_i \in [-1, 1]$. We make a simplistic assumption that t_i and t_j are iid U(0,1), in which case the density of $t_j - t_i$ is f. Another reason for choosing the uniform distribution is that it is the least biased prior distribution, and corresponds to the fact that nothing else is known about T (that is, there is no particular T on the basis of which S_1 and S_2 are being compared).

$$\begin{aligned} I_1(a,\gamma) &= \int_0^{\gamma a} (1-x)dx \\ &= \begin{cases} \int_0^1 (1-x)dx & \text{if} \quad \gamma a \geq 1 \\ \int_0^{\gamma a} (1-x)dx & \text{o.w.} \end{cases} \\ &= \begin{cases} \frac{1}{2} & \text{if} \quad \gamma \geq \frac{1}{a} \\ \gamma a - \frac{1}{2}\gamma^2 a^2 & \text{o.w.} \end{cases} \end{aligned} \tag{6.12}$$

Also, one may note that

$$\int_{\gamma a}^0 (1+x)dx = I_1(-a,\gamma).$$

Thus $D_\gamma(S_1, S_2; i, j)$ may be evaluated as

$$\begin{aligned} \int_{\gamma a_{ij}}^{\gamma b_{ij}} f(x)dx &= \begin{cases} \int_0^{\gamma b_{ij}} (1-x)dx - \int_0^{\gamma a_{ij}} (1-x)dx & \text{if } 0 \leq a_{ij} \leq b_{ij} \\ \int_0^{\gamma b_{ij}} (1-x)dx + \int_{\gamma a_{ij}}^0 (1+x)dx & \text{if } a_{ij} \leq 0 < b_{ij} \\ \int_{\gamma a_{ij}}^0 (1+x)dx - \int_{\gamma b_{ij}}^0 (1+x)dx & \text{if } a_{ij} \leq b_{ij} \leq 0 \end{cases} \\ &= \begin{cases} I_1(b_{ij},\gamma) - I_1(a_{ij},\gamma) & \text{if } 0 \leq a_{ij} \leq b_{ij} \\ I_1(b_{ij},\gamma) + I_1(-a_{ij},\gamma) & \text{if } a_{ij} \leq 0 \leq b_{ij} \\ I_1(-a_{ij},\gamma) - I_1(-b_{ij},\gamma) & \text{if } a_{ij} \leq b_{ij} \leq 0 \end{cases} \end{aligned} \tag{6.13}$$

It may be noted that we do not need to consider the case $a_{ij} \leq b_{ij} \leq 0$, since, $b = \max\left\{d_{ij}^{(1)}, d_{ij}^{(2)}\right\} \geq d_{ij}^{(1)} \geq 0$, by Assumption 1.

6.3.3 Characteristics and Discussion

It may be easily verified that the proposed measure for comparing two scoring systems, $D_\gamma(S_1, S_2)$ is a pseudometric. It is a metric when $\gamma \leq 1$. The following theorem provides a more general proof, where f is chosen to be any continuous probability density function on $(-1, 1)$ (i.e., taking strictly positive values on $(-1, 1)$, and 0 elsewhere), and is not restricted to the triangular density function.

Theorem 2 (Metric properties of $D_\gamma(S_1, S_2)$) *Let S_1 and S_2 be two normalized score vectors of length n, and let γ be a positive real number. Let the discordance for the pair (i, j) w.r.t. S_1 and S_2, $D_\gamma(S_1, S_2; i, j)$, be defined as in Eq. 6.10. where a_{ij} and b_{ij} are as defined in Eq. 6.8 and let $D_\gamma(S_1, S_2)$ be defined as in Eq. 6.9. Then, $D_\gamma(S_1, S_2)$ is a metric if $\gamma \leq 1$ and a pseudometric otherwise.*

Proof: To prove the theorem, it needs to be shown that $D_\gamma(S_1, S_2)$ satisfies the following properties:

$$D_\gamma(S_1, S_2) \geq 0 \ \forall\, S_1, S_2 \tag{6.14}$$

$$D_\gamma(S_1, S_1) = 0 \tag{6.15}$$

$$\text{If } \gamma \leq 1, D_\gamma(S_1, S_2) = 0 \Rightarrow S_1 = S_2 \tag{6.16}$$

$$D_\gamma(S_1, S_2) = D_\gamma(S_2, S_1) \tag{6.17}$$

$$D_\gamma(S_1, S_2) + D_\gamma(S_2, S_3) \geq D_\gamma(S_1, S_3) \tag{6.18}$$

Properties (6.14)-(6.17) may be easily verified from the definitions (6.9)–(6.11). Property (6.14) (non-negativity) follows immediately from the fact that each $D_\gamma(S_1, S_2; i, j)$ is the probability that a random variable with density f takes a value in a subinterval (a_{ij}, b_{ij}) of $(-1, 1)$. The proof of Property (6.15) is trivial, as each of the subintervals (a_{ij}, b_{ij}) are now of length 0.

To prove Property (6.16), we note that $\gamma \leq 1 \Rightarrow (\gamma a_{ij}, \gamma b_{ij}) \subseteq (-1, 1) \ \forall\, i < j$. Since $f(x) > 0$ if $x \in (-1, 1)$, we have

$$D_\gamma(S_1, S_2; i, j) = 0 \Leftrightarrow a_{ij} = b_{ij}.$$

So, $D_\gamma(S_1, S_2) = 0$ implies that

$$s_{1i} - s_{1j} = s_{2i} - s_{2j} \; \forall \; i < j \in \Omega.$$

Note that $s_{11} = 1$. Also,

$$\begin{aligned} & s_{21} - s_{2n} \\ = & \sum_{i=1}^{n-1} s_{2i} - s_{2,i+1} \\ = & \sum_{i=1}^{n-1} s_{1i} - s_{1,i+1} \\ = & \; s_{11} - s_{1n} \\ = & \; 1 \end{aligned}$$

Thus, $s_{21} - s_{2n} = 1$, and so the normalization constraint implies that $s_{21} = 1$ and $s_{2n} = 0$. Setting $j = i + 1$, and varying i from 1 to $n - 1$, we observe that $s_{1j} = s_{2j} \; \forall \; 2 \leq j \leq n$, and hence, $S_1 = S_2$. Thus Property (6.16) is proved. This property need not hold for $\gamma > 1$ because the integration interval may have no intersection with $(-1, 1)$, in which case, f would be 0 throughout the interval.

From the definition in Eq. 6.8, and the symmetry of $\max$ and $\min$,

$$a_{ij}^{S_1,S_2} = a_{ij}^{S_2,S_1} \text{ and } b_{ij}^{S_1,S_2} = b_{ij}^{S_2,S_1}, \; \forall \; i < j \in \Omega,$$

and hence,

$$D_\gamma(S_1, S_2; i, j) = D_\gamma(S_2, S_1; i, j),$$

thereby confirming Property (6.17).

To prove Property (6.18) we make use of the following facts about $\max$ and $\min$:

$$\min\{a, b\} \leq \max\{b, c\}, \tag{6.19}$$

$$\min\{a, c\} \geq \min\left\{\min\{a, b\}, \min\{b, c\}\right\} \tag{6.20}$$

and

$$\max\{a,c\} \leq \max\left\{\max\{a,b\}, \max\{b,c\}\right\}. \tag{6.21}$$

The fact (6.19) implies that

$$a_{ij}^{S_1,S_2} \leq b_{ij}^{S_2,S_3}, \text{ and } a_{ij}^{S_2,S_3} \leq b_{ij}^{S_1,S_2}, \tag{6.22}$$

and hence, the following inequalities hold:

$$\begin{aligned} & \left(a_{ij}^{S_1,S_2}, b_{ij}^{S_1,S_2}\right) \cup \left(a_{ij}^{S_2,S_3}, b_{ij}^{S_2,S_3}\right) \\ = & \left(\min\left\{a_{ij}^{S_1,S_2},\ a_{ij}^{S_2,S_3}\right\}, \max\left\{b_{ij}^{S_1,S_2},\ b_{ij}^{S_2,S_3}\right\}\right) \\ \supseteq & \left(a_{ij}^{S_1,S_3}, b_{ij}^{S_1,S_3}\right). \end{aligned} \tag{6.23}$$

Here, the first equality is a consequence of Eq. 6.22 which ensures that the union of the given intervals is indeed an interval. The second inequality is a consequence of Eqs. (6.20) and (6.21). It may be noted that the same inequalities hold even when the a_{ij}'s and b_{ij}'s are multiplied by a positive constant γ.

Integrating the non-negative function f over the above intervals (scaled by the constant γ), we have the following set of inequalities:

$$\begin{aligned} & \int_{\gamma a_{ij}^{S_1,S_2}}^{\gamma b_{ij}^{S_1,S_2}} f(x)dx + \int_{\gamma a_{ij}^{S_2,S_3}}^{\gamma b_{ij}^{S_2,S_3}} f(x)dx \\ \geq & \int_{\gamma \min\left\{a_{ij}^{S_1,S_2},\ a_{ij}^{S_2,S_3}\right\}}^{\gamma \max\left\{b_{ij}^{S_1,S_2},\ b_{ij}^{S_2,S_3}\right\}} f(x)dx \\ \geq & \int_{\gamma a_{ij}^{S_1,S_3}}^{\gamma b_{ij}^{S_1,S_3}} f(x)dx, \end{aligned} \tag{6.24}$$

which is the same as the triangular inequality

$$D_\gamma(S_1,S_2;i,j) + D_\gamma(S_2,S_3;i,j) \geq D_\gamma(S_1,S_3;i,j)$$

Summing over all pairs $(i<j)$, we have Property (6.18).

Thus, $D_\gamma(S_1,S_2)$ is a metric if $\gamma \leq 1$, and a pseudometric otherwise.

□

Kendall distance corresponds to a special case of the proposed metric by choosing f to have equal mass on $(-1, 0)$ and $(0, 1)$ (a weaker condition than symmetry), and β to be very close to zero, or equivalently, γ very large. Intuitively, this means that there is no fusion, and S_1 and S_2 are being compared directly to each other. It may be observed that, in such a case, the interval $(\gamma a_{ij}, \gamma b_{ij})$ either contains the whole of $(-1, 1)$ (when $a_{ij} < 0$) or does not have any intersection with $(-1, 1)$ (when $a_{ij} > 0$), which is the support of f, and thus, the degree of discordance is either 1 or 0, respectively. When there is a tie (and it is not a double tie), one limit of the integral in Eq. 6.10 becomes zero, and the other limit is either larger than 1 or smaller than -1, and hence, the degree of discordance is $\frac{1}{2}$. For a double tie, the degree of discordance is 0 for any γ (indicating perfect concordance). Thus, when γ is very large, $D_\gamma(S_1, S_2; i, j)$ assumes the value of 1 whenever (i, j) is a discordant pair w.r.t. S_1 and S_2, 0 when it is a concordant pair, $\frac{1}{2}$ in case of a single tie, and 0 for a double tie, and hence, $D_\gamma(S_1, S_2)$ is the Kendall distance between S_1 and S_2. Formally, this may be defined as:

$$K(S_1, S_2; i, j) = \lim_{\gamma \to \infty} D_\gamma(S_1, S_2; i, j) \; \forall \; i < j \tag{6.25}$$

For each pair, $\{i, j\}$, as γ increases, the value of $D_\gamma(S_1, S_2; i, j)$ monotonically decreases to 0 if $\{i, j\}$ is concordant, and increases to 1 otherwise. However, this does not imply that $D_\gamma(S_1, S_2)$ either monotonically increases or decreases with γ, the reason being that some of the individual components may increase while others decrease, and the rates may not balance each other. It may be easily verified in the case of $n = 3$, that $D_\gamma(S_1, S_2)$ may first increase and then decrease with γ.

$D_\gamma(S_1, S_2)$ is bounded above by $\frac{n(n-1)}{2}$. This is obvious because $D_\gamma(S_1, S_2; i, j)$ is bounded above by 1 for each pair $\{i, j\}$. Also, $D_\gamma(S_1, S_2)$ and $K(S_1, S_2)$ do not dominate each other. For example, when $K(S_1, S_2) = \frac{n(n-1)}{2}$, $D_\gamma(S_1, S_2)$ may be smaller (say, when $\gamma = 1$), and $D_\gamma(S_1, S_2)$ may be positive when $K(S_1, S_2)$ is zero (when $S_1 \neq S_2$ but $R(S_1) = R(S_2)$). Thus, $D_\gamma(S_1, S_2) - K(S_1, S_2)$ may be positive for certain choices of S_1 and S_2 and negative for some others.

Though the Kendall distance may be computed naïvely in $O(n^2)$ time, Knight's algorithm [74] based on mergesort, achieves the same in $O(n \log n)$ time by taking advantage of the redundancy involved in computing $K(S_1, S_2)$. No such algorithm is known, as yet, for computing $D_\gamma(S_1, S_2)$. Preliminary ongoing research in this direction is promising and suggests the existence of linear time algorithms to compute a "reasonable" approximation to $D_\gamma(S_1, S_2)$. Alternatively, when the number of items, n, is large, one may resort to a method like the one suggested by Fagin, et al. [48], where only the k top ranked items of both lists are considered for comparison. A "top k" version of the proposed metric is presented in the next section.

6.4 Comparing Top k Scores

A top k list is the set of items with the largest scores. Top k lists differ from full lists because two lists need not have the same set of items. If C is the set of items common to both the lists, then there are a total of $2k-|C|$ items in the two lists combined together, and thus, there are a total of $\frac{(2k-|C|)(2k-|C|-1)}{2}$ pairs. To compute the degree of discordance of a pair $\{i, j\}$, the four score values, viz, s_{1i}, s_{1j}, s_{2i} and s_{2j} need to be known. However, all four scores are known for only $\frac{|C|(|C|-1)}{2}$ pairs, and for the remaining pairs, either one or two of the scores are unknown, and hence, some sort of estimation needs to be performed for determining their degree of discordance.

We extend our procedure to comparing the top k scores of two scoring functions by mimicking the work of Fagin, et al. [48]. Fagin, et al. compared the top k lists obtained by two different rankings [48]. When dealing with items which appear in only one list, the definitions of ranks and discordance are appropriately modified, resulting in, among many others, a Kendall distance for top k lists.

6.4.1 Comparing Top k Lists

We first study the approach of Fagin, et al. [48] for generalizing the definition of discordance to the case of top k lists. We reproduce the text from [48] and, simultaneously, make a note of how the same extension for the computing the degree of discordance would differ in each case. Let τ_1 and τ_2 be two top k lists. The generalized discordance between i and j, w.r.t. two lists τ_1 and τ_2 is denoted $K^{(p)}(\tau_1, \tau_2; i, j)$.

Case 1 (i and j appear in both top k lists): If i and j are in the same order (such as i being ahead of j in both top k lists), then let $K^{(p)}(\tau_1, \tau_2; i, j) = 0$; this corresponds to "no penalty" for $\{i, j\}$. If i and j are in the opposite order (such as i being ahead of j in τ_1 and j being ahead of i in τ_2), then let the penalty $K^{(p)}(\tau_1, \tau_2; i, j) = 1$.

In this case, the usual definitions of discordance and degree of discordance are applicable.

Case 2 (i and j both appear in one top k list (say τ_1), and exactly one of i or j, say i, appears in the other top k list (τ_2)): If i is ahead of j in τ_1, then let the penalty $K^{(p)}(\tau_1, \tau_2; i, j) = 0$, and otherwise let $K^{(p)}(\tau_1, \tau_2; i, j) = 1$. Intuitively, we know that i is ahead of j as far as τ_2 is concerned, since i appears in τ_2 but j does not.

Here, there is no confusion regarding what the discordance should be as it is clear that i is ahead of j in τ_2. However, the degree of discordance needs the information regarding the separation between i and j. If i appears higher in τ_2, then, j is far below i as compared to when i is towards the bottom of τ_2. Also, since Fagin, et al. [48] consider only lists of items, there are no ties, whereas in our case, the scores may be tied.

Case 3 (i, but not j, appears in one top k list (say τ_1), and j, but not i, appears in the other top k list (τ_2)): Then let the penalty $K^{(p)}(\tau_1, \tau_2; i, j) = 1$. Intuitively, we know that i is ahead of j as far as τ_1 is concerned and j is ahead of i as far as τ_2 is concerned.

Again, though one is sure of discordance in this case, the degree of discordance may be partially inferred from the positions of i and j in their respective lists.

Case 4 (i and j both appear in one top k list (say τ_1), but neither i nor j appears in the other top k list (τ_2)): This is the interesting case (the only case where there is really an option as to what the penalty should be). Such pairs $\{i, j\}$ are called special pairs. In this case, we let the penalty $K^{(p)}(\tau_1, \tau_2; i, j) = p$.

This is the most difficult case, since the order of i and j in τ_2 is not known. However, the positions of i and j in τ_1 carries some information regarding what the degree of discordance may now be.

6.4.2 Degree of Discordance for Top k Scores

We shall now extend the definition of the degree of discordance to the case of the top k scores by making the maximum use of the available information and averaging out the unknown part. To compute the average over the unknown score values, we assume (as in Section 6.3.2) that they are uniformly distributed, and are independent of each other, and take the expectation. We assume that the top k scores of two scoring functions S_1 and S_2, say S_1^k and S_2^k, are given, along with the corresponding lists of items, τ_1 and τ_2. By Assumption 1, $\tau_1 = \{1, 2, \ldots, k\}$. Let $D_\gamma^k(S_1^k, S_2^k; i, j)$ denote the degree of discordance between i and j w.r.t. S_1^k and S_2^k (though not mentioned explicitly, $D_\gamma^k(S_1^k, S_2^k; i, j)$ involves τ_1 and τ_2 also).

Case 1 ($i, j \in \tau_1 \cap \tau_2$):

Since, $s_{1i}, s_{1j}, s_{2i}, s_{2j}$ are all known, the earlier definition is applied straightaway and $D_\gamma^k(S_1^k, S_2^k; i, j) = D_\gamma(S_1, S_2; i, j)$.

Case 2 ($i, j \in \tau_1$, but $i \in \tau_2$ and $j \notin \tau_2$):

So, s_{1i}, s_{1j}, s_{2i} are known but s_{2j} is unknown. All that is known about $y = s_{2j}$ is that $0 \leq s_{2j} \leq s_{2k} \leq s_{2i}$. Let a denote $s_{1i} - s_{1j}$. Therefore, we have

$$D_\gamma(S_1, S_2; i, j) = \int_{\gamma \min\{a, s_{2i} - y\}}^{\gamma \max\{a, s_{2i} - y\}} f(x)dx.$$

Since, y is unknown, we average the degree of discordance over all possible values of y by taking the expectation as follows:

$$\begin{aligned} D_\gamma^k(S_1^k, S_2^k; i, j) &= E\left[D_\gamma(S_1, S_2; i, j)\right] \\ &= \frac{1}{s_{2k}} \int_0^{s_{2k}} \int_{\gamma \min\{a, s_{2i}-y\}}^{\gamma \max\{a, s_{2i}-y\}} f(x)dx\, dy \end{aligned} \tag{6.26}$$

It may be noted that, as it is already given that $0 \leq y \leq s_{2k}$, the above is a conditional expectation, where y is assumed to be from $U(0, s_{2k})$ distribution. Also, Eq. 6.26 corresponds to the definition of $K^{(p)}(\tau_1, \tau_2; i, j)$ in Case 2 in Section 6.4.1. This may be seen by noting that when γ is large, the integral in Eq. 6.26 is 0, $\frac{1}{2}$ or 1 according as $a > 0$, $a = 0$ or $a < 0$.

Case 3 ($i \in \tau_1$, $j \in \tau_2$, and $i \notin \tau_2$, $j \notin \tau_1$):

Letting $y = s_{1j}$ and $z = s_{2i}$, and noting that $z - s_{2j} \leq 0 \leq s_{1i} - y$, the expected value of the degree of discordance may once again be computed as

$$\begin{aligned} D_\gamma^k(S_1^k, S_2^k; i, j) &= E\left[D_\gamma(S_1, S_2; i, j)\right[\\ &= \frac{1}{s_{1k} s_{2k}} \int_0^{s_{1k}} \int_0^{s_{2k}} \int_{\gamma(z-s_{2j})}^{\gamma(s_{1i}-y)} f(x)dx\, dz\, dy \end{aligned} \tag{6.27}$$

Case 4 ($i, j \in \tau_1$, $i, j \notin \tau_2$):

Let us denote $s_{1i} - s_{1j}$, s_{2i} and s_{2j} by a, y and z, respectively, and without loss of generality, assume that $a \geq 0$. Thereby, the expected degree of discordance is given by:

$$\begin{aligned} D_\gamma^k(S_1^k, S_2^k; i, j) &= E\left[D_\gamma(S_1, S_2; i, j)\right[\\ &= \frac{1}{s_{2k}^2} \int_0^{s_{2k}} \int_0^{s_{2k}} \int_{\min\{\gamma a, \gamma(y-z)\}}^{\max\{\gamma a, \gamma(y-z)\}} f(x)dx\, dz\, dy \end{aligned} \tag{6.28}$$

6.5 Applications

Scores contain more information than ranks, especially because the ranks may themselves be derived from the scores. So, comparing scorings finds applications in any field where

rankings need to be compared. We describe two such application areas related to page ranking. In addition, we elaborate on how the scores may also be used to measure how representative the ranks are.

6.5.1 Comparing Web Page Rankings

Ranking web pages has attracted the attention of several researchers, mainly due to the challenges it poses in terms of scalability and the imprecise and subjective nature of the task. Given the wide variety of ranking methods available, it is natural to compare them to decide which one is better. A more fundamental task is to decide whether the two given rankings are indeed different, and if so, how well-separated they are.

Though the task is to *rank* web documents, page ranking algorithms assign scores to pages. These scores are called page ranks. Existing works [20] compare the rankings by converting the scores into ranks and then computing the distance between these ranks. As discussed earlier in this chapter, and also, as is evident from the literature, these scores are seldom used in isolation for producing the final rankings. In such a case, the proposed methodology is more appropriate for comparing the page ranks, as it takes maximum advantage of the available information regarding the intended use of the page ranks. For example, if the fusing proportions (and thereby, γ) are known beforehand, the distance $D_\gamma(S_1, S_2)$ may be computed appropriately. On the other hand, if the algorithms produce the final ranking without fusing the scores, then, γ may be set to a very high value, which results in the computation of the Kendall distance.

It is common in the case of ranking web pages that the number of items is very large, and under such circumstances, $D^k_\gamma(S^k_1, S^k_2)$ should be used to compare S_1 and S_2 in terms of their top k scores, with k set to a few hundred or thousand.

6.5.2 Stopping Criterion for the Power Method

Another application of the proposed metric is in deciding when to stop the iterations in the *power method* [121]. The power method is used to obtain the dominant (or principal) eigenvector of matrix A, starting with an arbitrary vector $\mathbf{x}^{(0)}$. It is an iterative procedure, whereby successive vectors $\mathbf{x}^{(i+1)}$ are produced by multiplying A with $\mathbf{x}^{(i)}$, and is guaranteed to converge as the number of iterations tends to infinity.

In several studies like [11,82], the page rank vectors correspond to the principal eigenvectors of some transition probability matrix, and are computed iteratively by the power method. Once again, as in Section 6.5.1, the size of the document collection under consideration may be huge, in which case, each iteration is very costly, sometimes taking several hours to a few days [70], and hence, early stopping is desirable. The page rank vectors produced by consecutive iterations are compared to see if near convergence is attained.

In some instances, the computation is performed for a fixed number of iterations, say 50 or 100 iterations [11]. Though convergence may not be attained (in the L_1 or L_2 sense) by the time the computation is stopped, the resultant vector is declared to be the final page rank vector. The justification provided for such a behavior is that this vector serves its purpose in terms of ranking the documents, which is the final objective. In other words, even if the iterations are allowed to run for longer, the ranking would not change by much, as determined by the Kendall distance. Alternatively, one may base the stopping criterion in terms of the L_1 distance between the consecutive page rank vectors. However, since the ultimate objective is to rank the pages, it is preferable to use a rank comparison method for determining the stopping time [11].

While such a justification is acceptable if ranking is the sole objective, there might be other objectives too. In most cases, the page rank vector is considered as a set of importance scores and is combined with other entities, such as relevance scores, before the final ranking is produced [123]. Thus, it is natural to ask if it is sufficient to stop the computation after a certain number of iterations. The proposed distance measure may be used to check if (near) convergence has been attained. This convergence would be in a

sense that considers the purpose of computing the eigenvector.

6.5.3 Quantitative Measurement of the Representation of Scores by Ranks

A system ranking items on the basis of scores assigned to them, may choose to reveal only the ranks of those items, usually, by returning them in a particular order. While it is true that the items would be ordered in exactly the same manner on the basis of their scores too, the scores are only partially revealed. For example, let there be four items $(1, 2, 3, 4)$, each of which is assigned a score s_i, $i = 1, 2, 3, 4$, and the items are ordered in descending order of scores. If it is known that the ordered list is $1, 2, 3, 4$, then all that is revealed about the scores is that $1 = s_1 \geq s_2 \geq s_3 \geq s_4 = 0$. However, there is a general (human) tendency to perceive that the scores are uniformly distributed. So, the scores are implicitly assumed to be $1, \frac{2}{3}, \frac{1}{3}$ and 0, respectively (or something similar). This also corresponds to the average case, where one may observe that though s_3 may be either less than or greater than $\frac{1}{3}$, it is expected (under the assumption of uniformity) to be close to $\frac{1}{3}$. Such uniform scores, implicitly assumed on the basis of the ranks, shall be called the *uniformly perceived scores*, and the uniformly perceived score vector shall be denoted by $R(S)$ (or simply by R if S is clear from the context). For the sake of notational simplicity, we shall sometimes refer to uniformly perceived scores as just perceived scores in the rest of this paper.

In reality, the underlying scores may not be reflected properly by the rankings available. For instance, in the above example, the actual scores could have been $(1, 0, 0, 0)$, or $(1, 1, 1, 0)$ or $(1, 0.5, 0.5, 0)$, or any such 4-tuple satisfying the ordering criterion. So, the perceived scores may or may not reflect the underlying scores. Thereby, one should be able to measure if at all the perceived scores are similar to the actual scores.

The present work provides a methodology to quantify the separation between the actual scores and the perceived scores. Let there be n items $1, 2, \ldots, n$, and let their per-

ceived and actual scores be denoted by $r_1, r_2, \ldots, r_n$ and $s_1, s_2, \ldots, s_n$, respectively. So, the perceived score of the i^{th} item is given by $r_i = 1 - \frac{i-1}{n-1} = \frac{n-i}{n-1}$. It may be noted that Kendall distance between the scores and the perceived scores is zero, which is due to the fact that both of them result in the same rankings.

The degree of discordance of a pair $(i < j)$ is thereby given by

$$D_\gamma(R, S; i, j) = \int_{\min(s_i - s_j, \frac{j-i}{n-1})}^{\max(s_i - s_j, \frac{j-i}{n-1})} (1 - x)dx,$$

since, it is already known that $s_i \geq s_j$, and $r_j - r_i = \frac{j-i}{n-1} > 0$. If $s_i - s_j = \frac{j-i}{n-1}$, then the degree of discordance is zero. The other extreme is the case when the degree of discordance for the pair (i, j) is maximum. This happens, when $s_i - s_j$ is either 0 or 1 — which of the two is determined by $j - i$. If $j - i$ is close to $n - 1$, then, $D_\gamma(R, S; i, j)$ is maximized at $s_j = s_i$, whereas if $j - i$ is near 1, $s_i = 1$ and $s_j = 0$ maximizes the value of $D_\gamma(R, S; i, j)$.

As earlier, we would also be concerned with the total score based discordance between the two scorings. This is obtained by summing the degree of discordance over all possible pairs, and is given by

$$D_\gamma(R, S) = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} D_\gamma(R, S; i, j)$$

Again, if $s_i = 1 - \frac{i-1}{n-1}$, for each $i = 1, 2, \ldots, n$, the s_i's coincide with the r_i's and hence, $D_\gamma(R, S)$ turns out to be zero.

Since each $D_\gamma(R, S; i, j)$ is bounded above by $\frac{1}{2}$, it may be trivially seen that $D_\gamma(R, S) \leq \frac{n(n-1)}{4}$. However, finding R and S such that $D_\gamma(R, S)$ attains a maximum is not as simple as maximizing $D_\gamma(R, S; i, j)$ for each pair (i, j), the reason being that the pairs are not independent of each other, due to the monotonicity and normalization constraints. For example, if $n = 5$, $D_\gamma(R, S; 1, 2)$ is maximized when $s_2 = 0$ whereas $D_\gamma(R, S; 1, 3)$ is maximized when $s_3 = 1$, however, both cannot happen simultaneously (because $s_2 \geq s_3$).

How well R represents a particular score vector S_0 may be measured by the notion of the p-value of $D_\gamma(R, S_0)$. Consider the set, $\mathcal{S}_0$, of all S vectors such that $R(S) = R(S_0)$.

The p-value is the probability of having an S vector ($S \in \mathcal{S}_0$) such that $D_\gamma(R,S) \geq D_\gamma(R,S_0)$. In other words, the p-value of $D_\gamma(R,S)$ is the proportion of S vectors in $\mathcal{S}_0$ which are at an equal or higher distance from R than S_0 is from R. So, when most of the S vectors are such that $D_\gamma(R,S) \geq D_\gamma(R,S_0)$, then $D_\gamma(R,S_0)$ may be considered to be small, and *vice versa*. Thus, when the p-value corresponding to $D_\gamma(R,S)$ is very small, it may be declared that $R(S)$ does not represent S well enough.

6.6 Experimental Results

To understand the significance of the present work and to validate the claims made in this chapter, several experiments of the following kinds were conducted.

- Study of the behavior of $D_\gamma(S_1,S_2)$ for various values of γ.
- Study of the behavior of $D_\gamma^k(S_1,S_2)$ for various values of n and k, and testing the dependence on the assumption of uniformity.
- Determining the number of iterations for eigenvector computation.
- Predicting the discordance in a pair of vectors after score fusion.
- Computing the distance between uniformly perceived and actual scores.

We now describe each of these experiments and their results along with our observations and analysis.

6.6.1 Behavior of $D_\gamma(S_1,S_2)$

It is theoretically assured that $D_\gamma(S_1,S_2) \leq \frac{n(n-1)}{2}$, and also that $\lim_{\gamma\to\infty} D_\gamma(S_1,S_2) = K(S_1,S_2)$. Moreover, our remarks on Page 153 make it amply clear that $D_\gamma(S_1,S_2)$ is not a monotone function of γ although each $D_\gamma(S_1,S_2;i,j)$ is so.

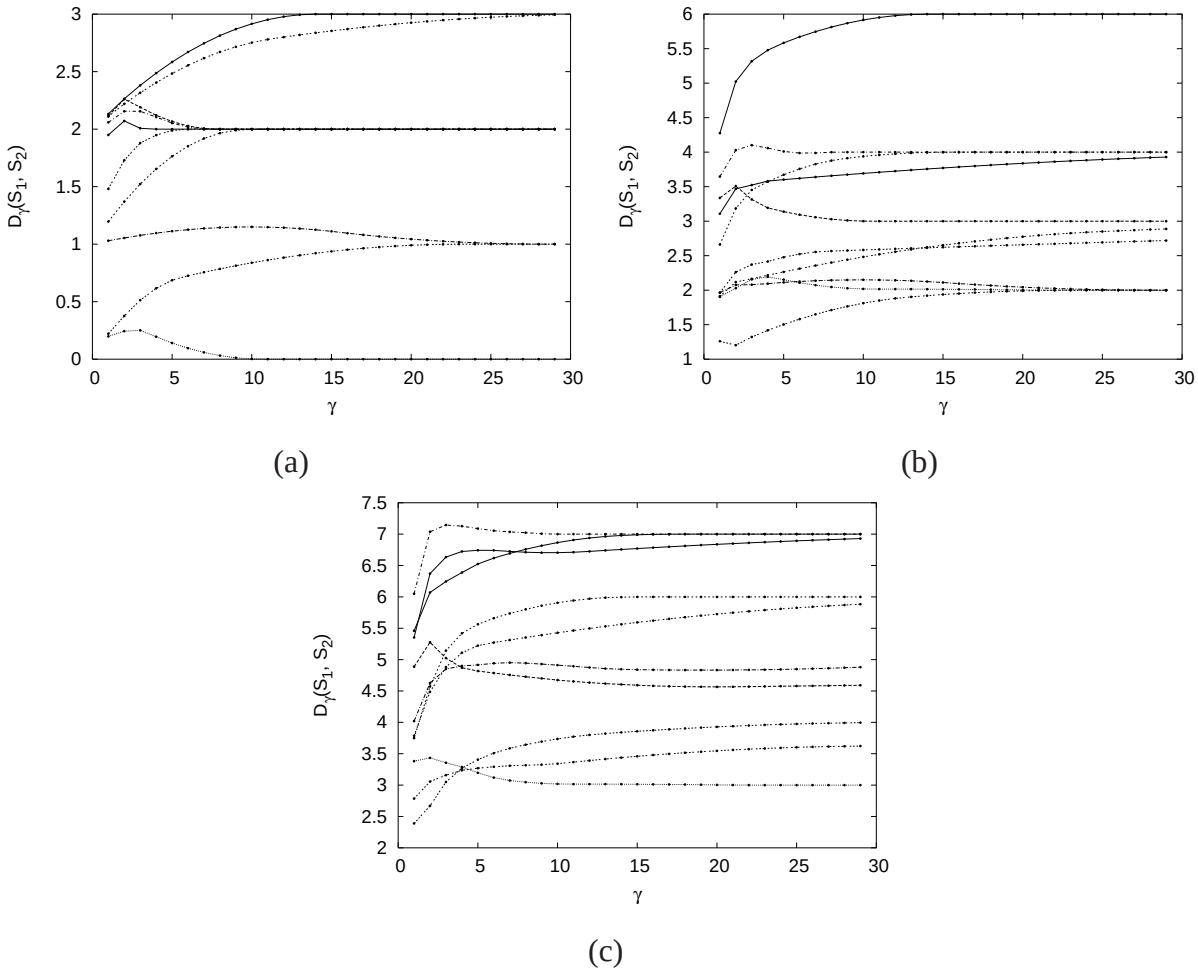


Figure 6.2: Plots of $D_\gamma(S_1, S_2)$ vs. γ, for 10 randomly chosen (S_1, S_2) pairs, with (a) $n = 3$, (b) $n = 4$ and (c) $n = 5$

The aforementioned properties are graphically depicted in Figs. 6.2(a), 6.2(b) and 6.2(c). Here, n is set to $3, 4$ and 5, respectively, and ten S_1, S_2 pairs are randomly generated, and $D_\gamma(S_1, S_2)$ is computed for values of γ varying from 1 to 30. The Kendall distance, $K(S_1, S_2)$ may take values only from $\{0, 1, \ldots \frac{n(n-1)}{2}\}$, and the $D_\gamma(S_1, S_2)$ values are seen to be converging to the respective Kendall distances. The rates of convergence, however, are different in each case. Also, in some of the cases, $D_\gamma(S_1, S_2)$ varies monotonely with γ, whereas, in the remaining cases, it is not so. However, eventually (beyond some value of γ), monotonicity is restored in each of the cases.

6.6.2 Behavior of $D^k_\gamma(S_1, S_2)$

We now study the properties of $D^k_\gamma(S_1, S_2)$ as k varies from 1 to n. A pair of score vectors is generated randomly and min-max normalization [84] is applied. The top k items (according to the scores) are selected from each vector, while the scores of the remaining items are assumed to be unknown, and $D^k_\gamma(S_1, S_2)$ is computed as described in Section 6.4.2. Since $D^k_\gamma(S_1, S_2)$ is an expected value, with the expectation being taken over all the unknown score values, it is imperative to know how good this approximation is. In the present case, all the score values are known, and therefore, the exact value $E^k_\gamma(S_1, S_2)$ may also be computed by summing up the (exact) degree of discordance of all the pairs appearing in the union of the two top k lists.

The computed values of $D^k_\gamma(S_1, S_2)$, $E^k_\gamma(S_1, S_2)$ and $D^k_\gamma(S_1, S_2) - E^k_\gamma(S_1, S_2)$ are shown graphically in Fig. 6.3, from which it may be seen that the $D^k_\gamma(S_1, S_2)$ approximates $E^k_\gamma(S_1, S_2)$ very well. It may be noted that the computation of $D^k_\gamma(S_1, S_2)$ is based on the assumption that S_1 and S_2 arise from the Uniform ($U(0, 1)$) distribution. In order to test the dependence of the approximation on the distributional assumptions, two more experiments were conducted, with S_1 and S_2 arising from the Gaussian ($N(0, 1)$) distribution in one and the Exponential ($E(1)$) distribution in the other. The results are presented in Figs. 6.4 and 6.5, respectively, and it may be observed that there is a consistent over-estimation in the case of the Gaussian distribution, whereas, the approximation

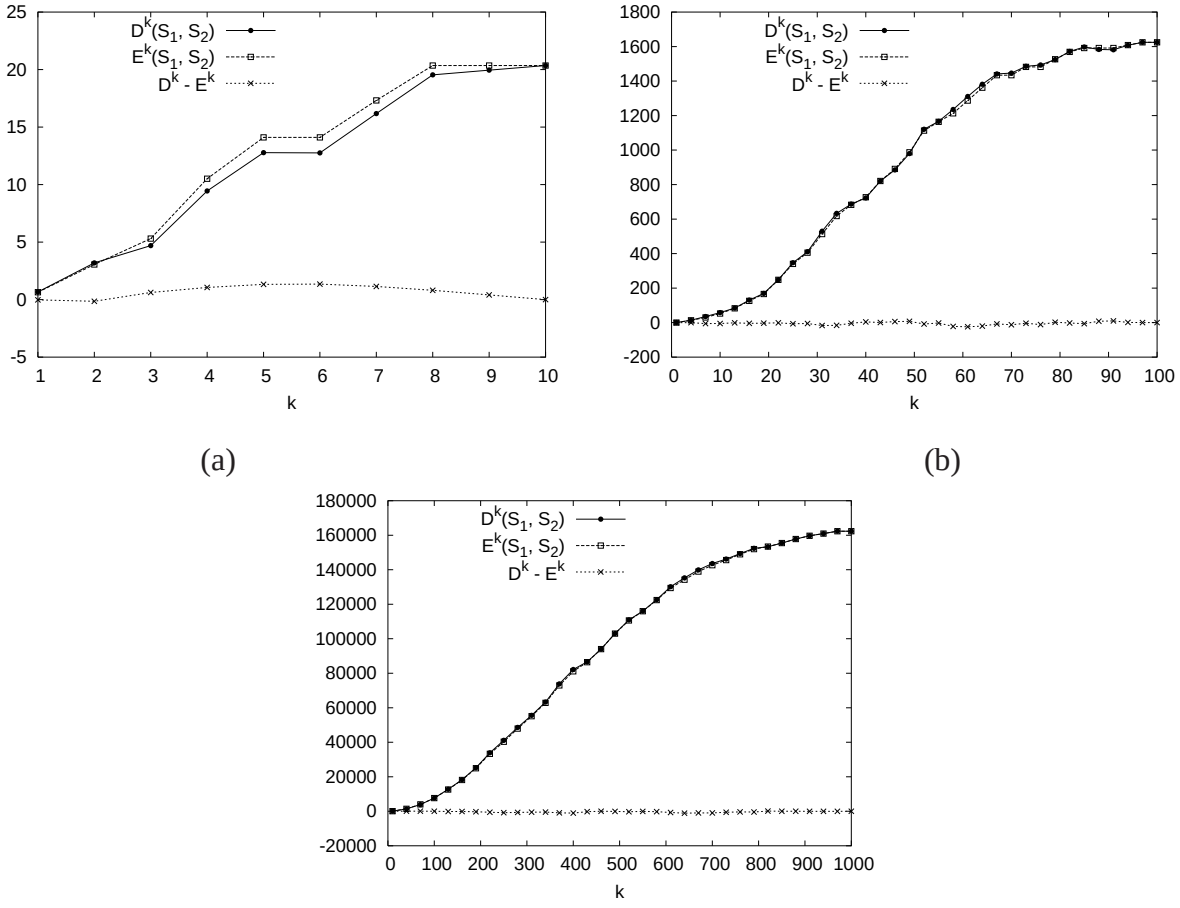


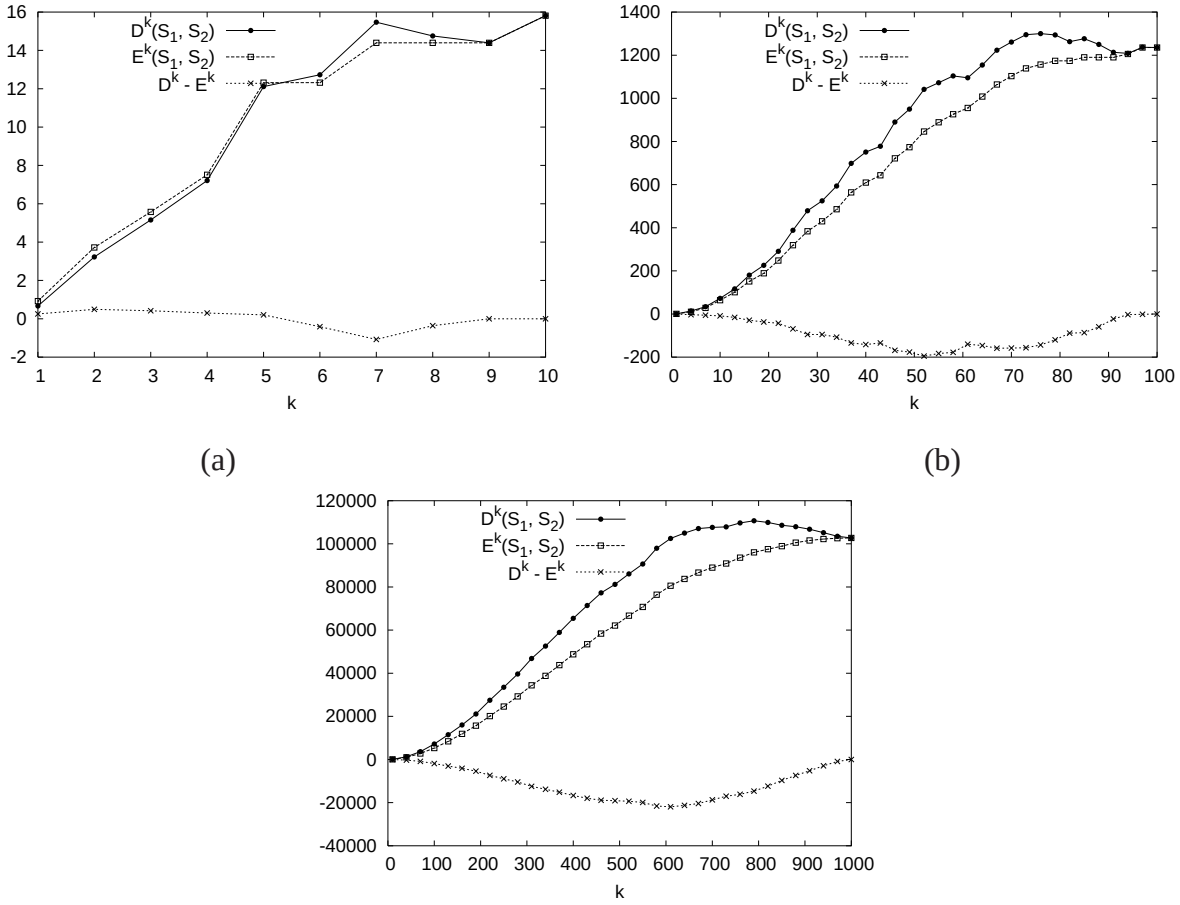
Figure 6.3: Plots of $D^k(S_1, S_2)$, $E^k(S_1, S_2)$ and $D^k(S_1, S_2) - E^k(S_1, S_2)$ vs. k, for (S_1, S_2) generated from Uniform distribution, with (a) $n = 10$, (b) $n = 100$ and (c) $n = 1000$

is very good in the case of the Exponential distribution.

6.6.3 Determining the number of iterations for computing page ranks

As discussed in Section 6.5.2, one would like to know when to terminate the power iterations based on the amount of change in the rankings in the consecutive iterations. We chose two data sets from Stanford's WebBase [65] and named them WB1_7440 and WB4_7060, after the host and port numbers from which they are available. The former is a crawl of a part of the berkeley.edu domain, and there are about 140 thousand (140K) pages with over 1.6 million (1.6M) links to pages within the same data set. WB4_7060, which is a crawl of a part of the stanford.edu domain, consists of about 40 thousand (40K) pages and over 260 thousand (260K) links to pages within itself. The PageRank [60] algorithm was run for 100 iterations on both the chosen data sets. We then computed $K^{(0.5)}(S_i, S_{i+1})$ and $D^k(S_i, S_{i+1})$ (γ set to 1), which are the top k versions of Kendall distance and the proposed distance, respectively. Here S_i is the page rank vector at the end of the i^{th} iteration, and k was chosen to be 100, 1000 and 5000. We have also computed $K^{(p)}(S_i, S_{100})$ and $D^k(S_i, S_{100})$, though these quantities would not be available *during* the page rank computation. These values are presented in the plots in Figs. 6.6 and 6.7.

It may be noted from Figs. 6.6 and 6.7 that if $D^k(S_i, S_{i+1})$ is to be used instead of $K^{(0.5)}(S_i, S_{i+1})$, (near) convergence is declared much earlier. For example, in Fig. 6.6a, $D^k(S_i, S_{i+1})$ would have recommended the termination of the procedure after 20 iterations, whereas, $K^{(0.5)}(S_i, S_{i+1})$ would have led to at least 26 iterations. This indicates that once it is decided that the obtained ranks would be fused together with some other score vector (in equal proportions, since we have set γ to be 1), there would be no significant improvement by continuing beyond 20 iterations.



(a) (b)

(c)

Figure 6.4: Plots of $D^k(S_1, S_2)$, $E^k(S_1, S_2)$ and $D^k(S_1, S_2) - E^k(S_1, S_2)$ vs. k, for (S_1, S_2) generated from Gaussian distribution, with (a) $n = 10$, (b) $n = 100$ and (c) $n = 1000$

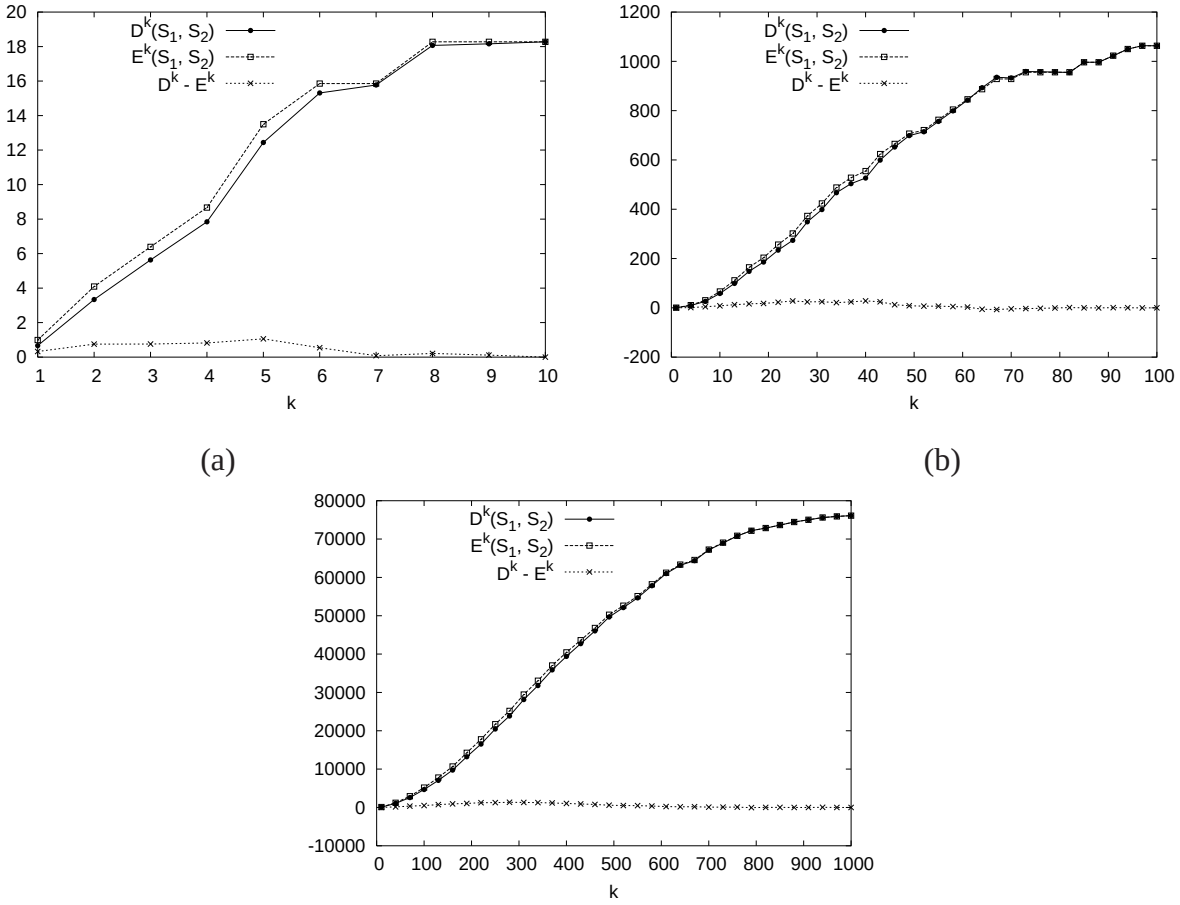


Figure 6.5: Plots of $D^k(S_1, S_2)$, $E^k(S_1, S_2)$ and $D^k(S_1, S_2) - E^k(S_1, S_2)$ vs. k, for (S_1, S_2) generated from Exponential distribution, with (a) $n = 10$, (b) $n = 100$ and (c) $n = 1000$

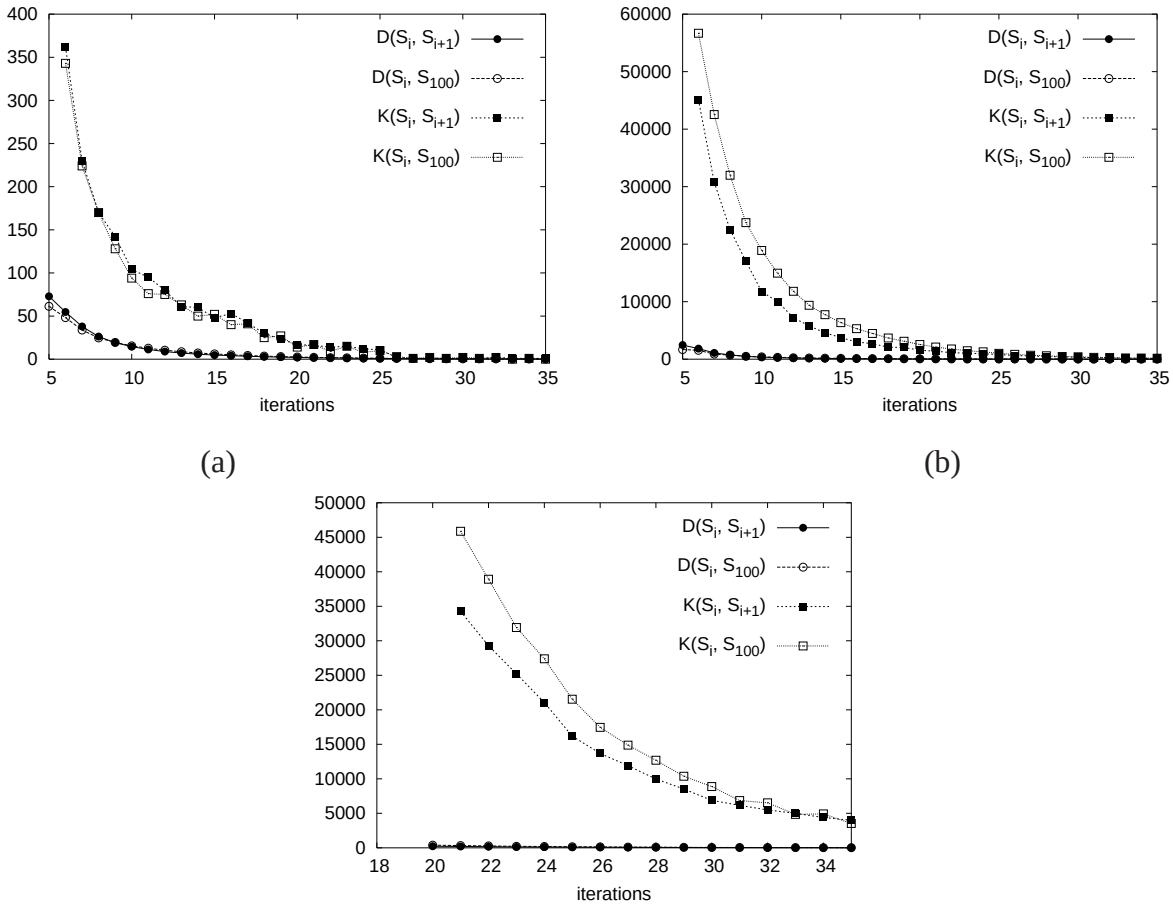


Figure 6.6: Plots of $D^k(S_i, S_{i+1})$, $D^k(S_i, S_{100})$, $K^{(0.5)}(S_i, S_{i+1})$, and $K^{(0.5)}(S_i, S_{100})$ vs. i, for the WB1_7440 data set with (a) $k = 100$, (b) $k = 1000$ and (c) $k = 5000$

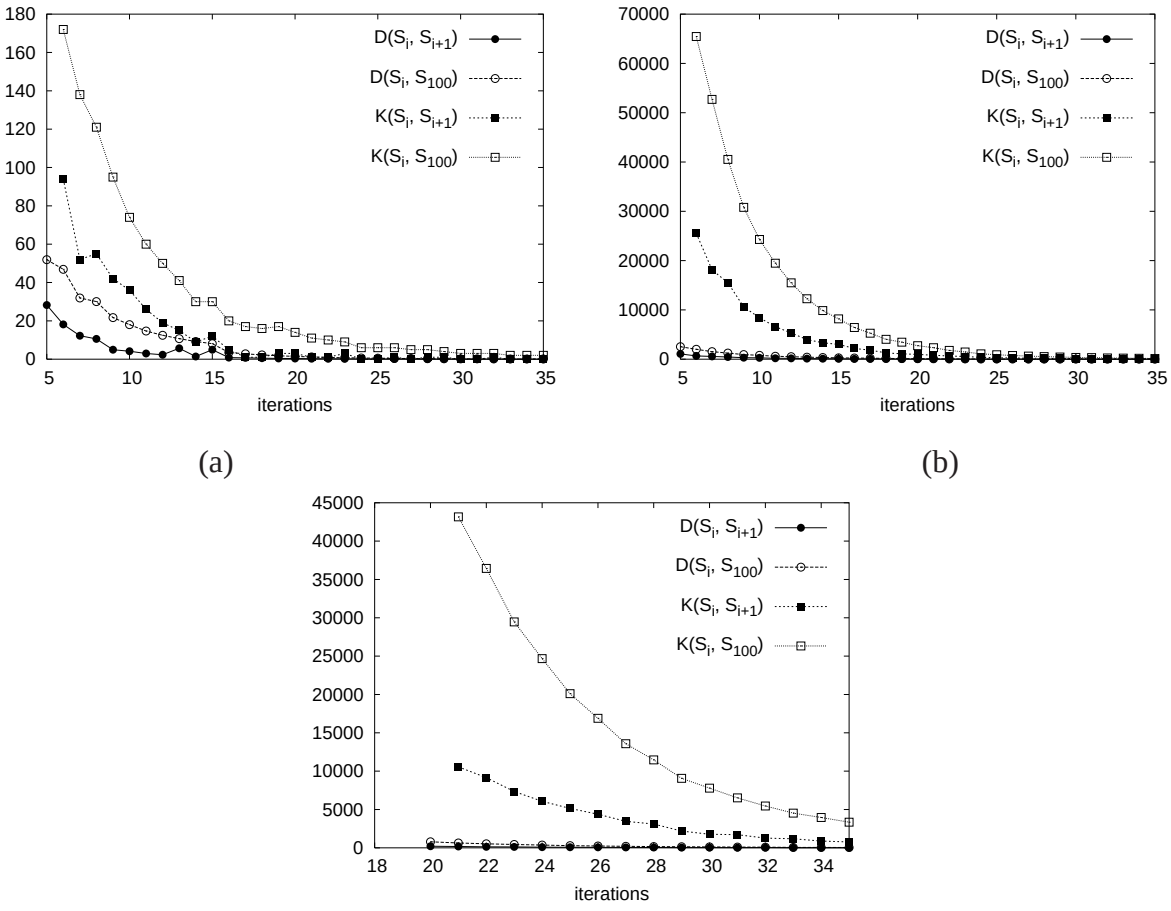


Figure 6.7: Plots of $D^k(S_i, S_{i+1})$, $D^k(S_i, S_{100})$, $K^{(0.5)}(S_i, S_{i+1})$, and $K^{(0.5)}(S_i, S_{100})$ vs. i, for the WB4_7060 data set with (a) $k = 100$, (b) $k = 1000$ and (c) $k = 5000$

6.6.4 Predicting discordance after score fusion

This set of experiments is aimed at predicting the discordance after score fusion, without actually performing the fusion. Such requirements arise often in the Web domain, say, for comparing page rank vectors. The traditional method of comparing page rank vectors is to use each of them to retrieve the top pages for a set of queries and computing the discordance between them (see, for example, [123]). This involves doing the following for each query. The set of documents matching (or containing) the query is identified, and the query relevance scores for each document are computed. These relevance scores are combined with the ranks of the documents, and the documents are ordered according to the fused scores. A rank comparison measure is then computed between the top k lists arising from each of the page rank vectors fused with the relevance vectors.

Ideally, the set of queries should be very large so that a comprehensive comparison between the page rank vectors may be made. In addition, the number of documents in the corpus may be huge, too. Under such circumstances, it is computationally prohibitive to compare the page rank vectors for various choices of the weights (or fusing parameters). To this end, the proposed measure may be employed to circumvent the actual computation of discordance measures between the fused score vectors by having a reasonable approximation as demonstrated by the following experiment.

We consider the WB1_7440 and WB4_7060 corpora once more. Two page rank vectors, labeled S_1 and S_2, over these data sets are obtained by considering the vectors after 3 and 50 iterations, respectively, of the eigenvector computation mentioned earlier. Note that the exact method of obtaining these vectors is not relevant to the present experiment. The fusing proportion β ($= 1 - \alpha$) is varied over the values 0.25, 0.50, and 0.75.

We first compare the two vectors in the following naive manner. We choose each word in the corpus as a single-term query, and stem them using Porter's algorithm [118]. An inverted index is created for each data set. Now, for each stem word w in the corpus, the list of documents containing that stem is extracted, and the TFIDF [127] vector, T_w, is computed. Let the corresponding page rank vectors, of the same length as T_w, be

called S_{1w} and S_{2w}, respectively. There are over 44 thousand (44K) and 35 thousand (35K) distinct stems in WB1_7440 and WB4_7060 data sets, respectively. When only the 5000 top ranked documents are considered in these datasets, there are over 14K and 16K distinct stems, respectively.

Now for each word, we compute $K^{(0.5)}(S_{1w}, S_{2w})$ and $K^{(0.5)}(\alpha S_{1w} + \beta T_w, \alpha S_{2w} + \beta T_w)$, which are the top k discordance values between S_{1w} and S_{2w}, and between the fused vectors $\alpha S_{1w} + \beta T_w$ and $\alpha S_{2w} + \beta T_w$, respectively.

These quantities are averaged over all the words by dividing their sum by the sum of all possible pairs for each word, and the averages are denoted $K\tau_{0.5}^k(S_1, S_2)$ and $D\tau_\gamma^k(S_1, S_2)$ (where, as earlier, $\gamma = \frac{1-\beta}{\beta}$), respectively, reflecting the quantities they estimate. Mathematically, if df_w is the number of documents in which w appears, the average is computed as

$$K\tau_{0.5}^k(S_1, S_2) = \frac{2\sum_w K^{(0.5)}(S_{1w}, S_{2w})}{\sum_w df_w(df_w - 1)}, \tag{6.29}$$

and

$$D\tau_\gamma^k(S_1, S_2) = \frac{2\sum_w K^{(0.5)}(\alpha S_{1w} + \beta T_w, \alpha S_{2w} + \beta T_w)}{\sum_w df_w(df_w - 1)}. \tag{6.30}$$

The above procedure is repeated using just the top 5000 documents in each data set. These averages, which would have been the measurements from the traditional rank comparison technique described earlier, are tabulated in Table 6.1.

Next, we employ the proposed metric to obtain an approximation of the values in Table 6.1. The discordance measures $K\tau_{0.5}^k(S_1, S_2)$ and $D\tau_\gamma^k(S_1, S_2)$ are shown in Table 6.2. Here, $K\tau_{0.5}^k(S_1, S_2)$ is the top k Kendall distance $K^{(0.5)}(S_1, S_2)$ normalized by all possible pairs of pages in the union. Similarly, $D\tau_\gamma^k(S_1, S_2)$ is the normalized version of $D_\gamma^k(S_1, S_2)$. These computations are repeated for the top 5000 pages of each data set, and are also tabulated in Table 6.2. Note that the values in Table 6.2 are independent of the content of the web pages, or to rephrase, the relevance of pages to queries is not considered during ranking.

We observe from Tables 6.1 and 6.2 that the corresponding values in Tables 6.1 and 6.2 are very similar and show similar patterns and trends. For example, on the WB1_7440

Table 6.1: Kendall and degree of discordance values averaged over all words

WB1_7440				WB4_7060			
Pages selected (k)	$K\tau_{0.5}^k$	γ	$D\tau_\gamma^k$	Pages Selected (k)	$K\tau_{0.5}^k$	γ	$D\tau_\gamma^k$
All (140K)	0.0160	3	0.0049	All (40K)	0.0222	3	0.0096
All (140K)	0.0160	1	0.0023	All (40K)	0.0222	1	0.0049
All (140K)	0.0160	$\frac{1}{3}$	0.0012	All (40K)	0.0222	$\frac{1}{3}$	0.0020
Top 5000	0.1220	3	0.0279	Top 5000	0.1130	3	0.0409
Top 5000	0.1220	1	0.0137	Top 5000	0.1130	1	0.0200
Top 5000	0.1220	$\frac{1}{3}$	0.0067	Top 5000	0.1130	$\frac{1}{3}$	0.0081

Table 6.2: Normalized Kendall and degree of discordance values

WB1_7440				WB4_7060			
Pages selected (k)	$K\tau_{0.5}^k$	γ	$D\tau_\gamma^k$	Pages Selected (k)	$K\tau_{0.5}^k$	γ	$D\tau_\gamma^k$
All (140K)	0.0189	3	0.0032	All (40K)	0.0253	3	0.0067
All (140K)	0.0189	1	0.0017	All (40K)	0.0253	1	0.0029
All (140K)	0.0189	$\frac{1}{3}$	0.0006	All (40K)	0.0253	$\frac{1}{3}$	0.0011
Top 5000	0.1214	3	0.0133	Top 5000	0.1090	3	0.0323
Top 5000	0.1214	1	0.0078	Top 5000	0.1090	1	0.0110
Top 5000	0.1214	$\frac{1}{3}$	0.0045	Top 5000	0.1090	$\frac{1}{3}$	0.0054

data set, when γ is set to 1, the discordance value as found from the naive experiment is 0.0023, and its estimate turns out to be 0.0017. When the possibility of fusion is not taken into consideration, the corresponding estimate would have been 0.0189, which is well away from 0.0023. The $D\tau_\gamma^k$ values in Table 6.2, however, are all underestimates of the corresponding values in Table 6.1, possibly due to the TFIDF scores not being uniformly distributed. Nevertheless, these values, differ significantly from Kendall's τ values, and shrink as β is increased. Moreover, it may be seen from Figs. 6.8 and 6.9 that as β increases, the shrinking happens for each word (and not just on average), as the green (lightly shaded) histograms are squeezed to the left. The blue (densely shaded) histograms remain stationary. The plots for the full document sets are similar, but are less informative by virtue of a much larger proportion of zeros (the leftmost bars).

In conclusion, the proposed metric helps us directly predict the discordance between the two page rank vectors instead of taking recourse to the traditional rank based comparisons which involve comparing fused vectors for a large number of queries which, in the present case, run into several thousands. In cases where more relevance factors are involved (for example, search engines like Yahoo! and Google weight hundreds of factors to compute relevance [43]), for comparing two variants of a particular factor, all that is needed to be known is the fusion parameter (α) for the factor under consideration. That way, without knowing the weights for the remaining factors involved, or even what the exact factors are, one may estimate the amount of discordance that could result from differences in the two variants of this single factor.

6.6.5 Uniformly Perceived versus Actual Scores

This experiment is aimed at measuring the similarity between the uniformly perceived and actual scores by computing the distance $D(R, S)$. In other words, this is an attempt to quantify how much information is lost by converting the scores to ranks. For this purpose, the Economic Freedom Index (EFI) data set is chosen, in which 20 states of India were assigned a set of composite scores, and were ranked accordingly [40]. The

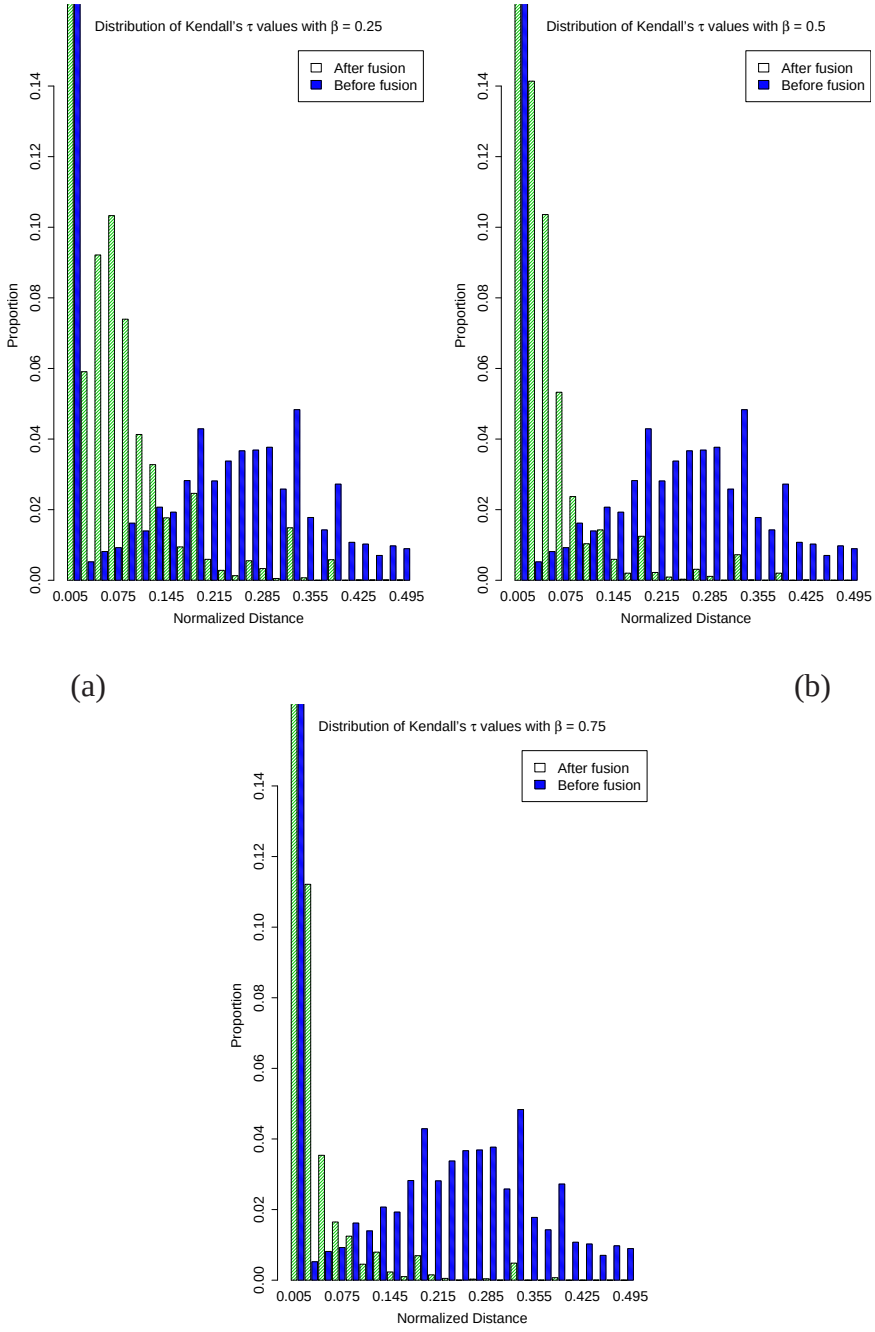


Figure 6.8: Histograms of Kendall's τ values for each word in the top 5000 pages of WB1_7440 data set with and without the TFIDF vectors fused to the rank vectors, and the fusion parameter β set to (a) 0.25, (b) 0.50 and (c) 0.75

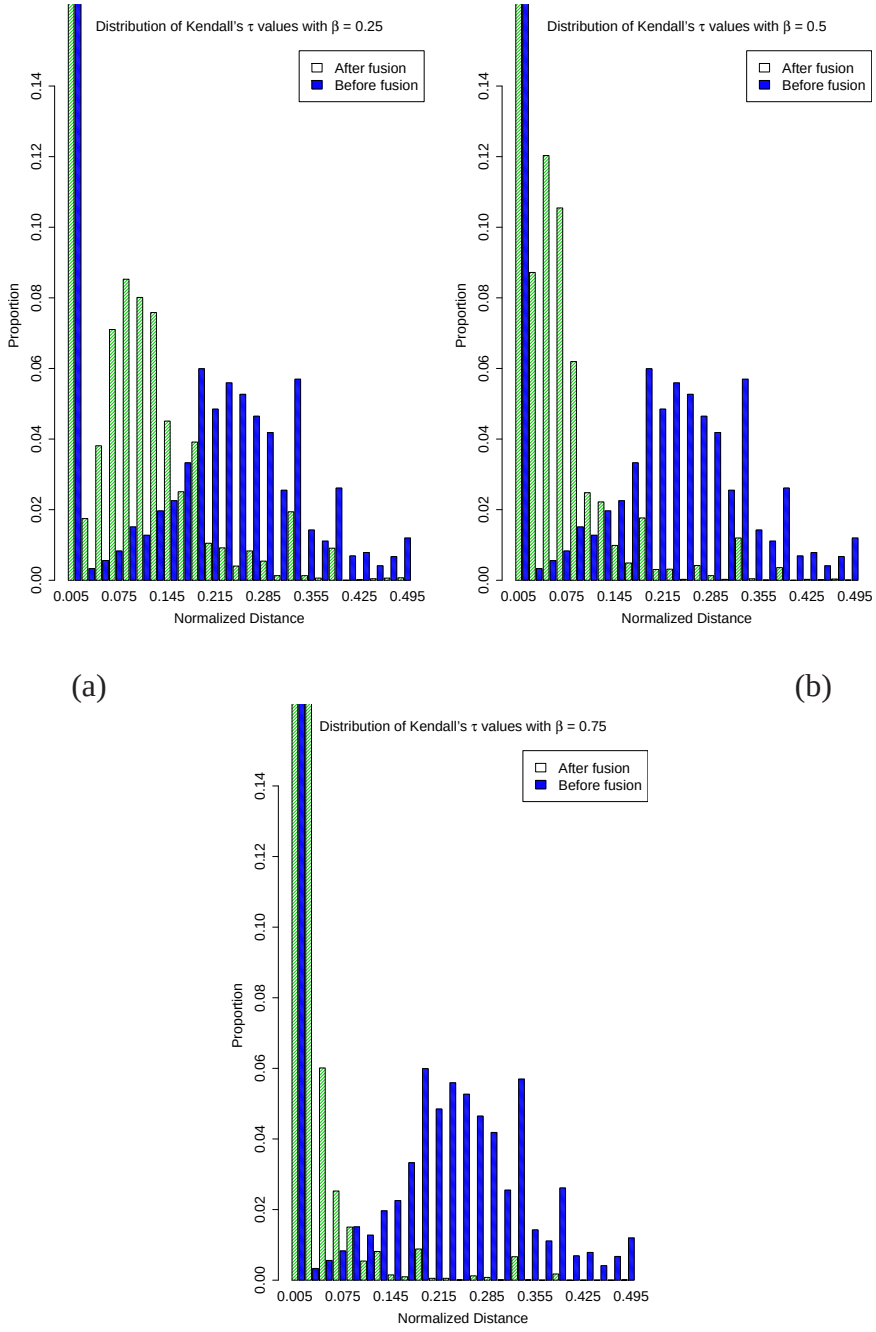


Figure 6.9: Histograms of Kendall's τ values for each word in the top 5000 pages of WB4_7060 data set with and without the TFIDF vectors fused to the rank vectors, and the fusion parameter β set to (a) 0.25, (b) 0.50 and (c) 0.75

Table 6.3: Economic Freedom Index for 20 States of India

Rank	State	EFI	Rank	State	EFI
1	Gujarat	0.40	11	Orissa	0.32
2	Andhra Pradesh	0.38	12	Karnataka	0.31
3	Kerala	0.37	13	Uttar Pradesh	0.30
4	Chhattisgarh	0.37	14	West Bengal	0.30
5	Tamil Nadu	0.37	15	Himachal Pradesh	0.30
6	Maharashtra	0.37	16	Jharkhand	0.29
7	Rajasthan	0.35	17	Punjab	0.29
8	Haryana	0.35	18	Uttaranchal	0.28
9	Madhya Pradesh	0.33	19	Bihar	0.26
10	Jammu & Kashmir	0.33	20	Assam	0.22

data set is presented in Table 6.3.

For the EFI dataset, the distance between ranks and scores is 11.36. In order to be able to judge how good or how bad it is, we generated several score vectors (of size 20) uniformly, and counted the number of times, a score vector had a distance of 11.36 or more from the perceived score vector. The p-value turns out to be 0.12, which indicates that only about 12% of score vectors are more separated from the perceived score vectors. This, in turn, signifies that the scores in the EFI data set are not very well represented by the corresponding ranks (through the perceived scores). The authors of [40] had expressed their opinion that the ranks did not represent the scores well. The results of our experiments now provide a quantitative evidence for the same.

The individual degrees of discordance for each pair of states are displayed in Table 6.4. We have also computed the p-values for each of the individual cells. For each pair of states, we count the number of randomly generated score vectors with a higher degree of discordance for the same pair of states. All entries with the corresponding p-value less

than 0.05 are shown in bold, and there are 31 such values, which is about 16% of the total of 190 entries. It may be noted that the 31 entries in bold are not the largest entries of Table 6.4. For example, the entry at position $(3, 8)$ (0.12) is larger than that at $(3, 19)$ (0.06), however, the former is more likely to occur than the latter.

6.7 Conclusions and Future Work

The present chapter dealt with generalizing measures of discordance for the case when the underlying scores are known. A metric has been provided to compare score vectors directly. This metric turns out to be the Kendall distance when a parameter γ, denoting the ratio of fusing proportions, is large. Experiments of various kinds demonstrate the wide range of theory and applications of the metric introduced in the present work.

There is a tremendous scope for future work, including studying the cases where T is assumed to arise from specific distributions, obtaining the properties such as maximum and minimum of $D_\gamma(S_1, S_2)$ and $D^k_\gamma(S_1, S_2)$ for particular values of γ, and speeding up the computation of the proposed metric.

Table 6.4: Degree of discordance between the actual and uniformly perceived scores for each pair of states in the EFI data set

	State																			
State	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Total
1	0.05	0.05	0.01	0.04	0.08	0.03	0.06	0.02	0.05	0.04	0.04	0.03	0.05	0.06	**0.05**	**0.06**	**0.05**	**0.02**	0	0.79
2		0	0.05	0.09	0.13	0.08	0.11	0.06	0.09	0.08	0.07	0.07	**0.09**	**0.10**	**0.09**	**0.10**	**0.09**	**0.05**	0	1.42
3			0.05	0.10	**0.15**	0.08	0.12	0.07	0.10	0.09	0.08	0.07	**0.10**	**0.12**	**0.10**	**0.12**	**0.10**	**0.06**	0.01	1.60
4				0.05	0.10	0.04	0.08	0.03	0.07	0.06	0.05	0.05	0.07	**0.10**	**0.09**	**0.10**	**0.09**	**0.05**	0	1.15
5					0.05	0.01	0.04	0.01	0.03	0.03	0.02	0.02	0.05	0.07	0.07	**0.09**	**0.08**	0.04	0.01	0.88
6						0.05	0.01	0.05	0.01	0.01	0.01	0.01	0.02	0.05	0.04	0.07	0.06	0.03	0.02	0.94
7							0.05	0.01	0.04	0.04	0.03	0.03	0.06	0.09	0.08	**0.11**	**0.10**	0.06	0.01	0.99
8								0.05	0.01	0.01	0.01	0.01	0.03	0.06	0.05	0.08	0.07	0.04	0.03	0.93
9									0.05	0.05	0.04	0.04	0.08	0.11	0.10	**0.13**	**0.12**	0.07	0.01	1.11
10										0	0.01	0.01	0.04	0.08	0.07	0.10	0.09	0.05	0.04	0.93
11											0	0.01	0.04	0.08	0.08	0.11	0.10	0.05	0.04	0.93
12												0	0.05	0.09	0.08	0.12	0.11	0.06	0.04	0.94
13													0.05	0.10	0.09	0.13	0.12	0.07	0.05	0.95
14														0.05	0.05	0.09	0.08	0.03	0.08	1.09
15															0	0.05	0.04	0.01	0.12	1.39
16																0.05	0.05	0.01	0.12	1.28
17																	0	0.05	**0.17**	1.76
18																		0.05	**0.18**	1.59
19																			**0.15**	0.96
20																				1.08

Chapter 7

Conclusions, Discussion and Scope for Further Work

In every chapter we have presented conclusions drawn from the respective methodologies developed and the experimental results therein. Here we consolidate them to provide an overall picture of the contributions of the thesis.

Chapters 2 and 3 described the groundwork (preprocessing of web data) that needs to be performed before the task of modeling a surfer may be taken up. These chapters involved preprocessing of the available hypertext data sets which may then be used for estimating the models provided in Chapters 4 and 5, and thereby, gain insights into a surfer's behavior. These insights may be utilized for tasks like page ranking and categorization. Chapter 6 deals with the analysis of different (page) ranking schemes.

In Chapter 2 we had described the design of a stemming algorithm which uses the classification information of a corpus to refine a given stemmer. We had introduced a procedure similar to sequential hypothesis testing to identify groups of words that would be stemmed to the same stem. The main advantage over other stemmers like co-occurrence based stemmers is its ability to drastically reduce the dictionary size while maintaining both the classification accuracy and retrieval precision. The superiority of the proposed methodology was experimentally demonstrated for the task of text categorization when

Naive Bayes classifier, Support Vector Machines and Maximum Entropy Method based classifier were used. This was also supported by precision-recall based evaluation. Another set of experiments performed on WSJ data set demonstrated the enhancement in retrieval precision when the refined stemmers were employed instead of existing stemmers. The performance of refinement done by employing the classification information from a different corpus increased as the number of common words increased.

Whereas Chapter 2 focused on preprocessing the textual contents of documents, Chapter 3 concentrated on preprocessing the documents based on the hyperlink structure. We have developed a novel methodology for the task of detecting sequences of web pages. Also, the importance of sequence detection has been highlighted extensively. With the help of some examples,we have explained why detecting *all* possible sequences (or cycles) of web pages is neither feasible nor interesting. Subsequently, we described the sequences of interest, and then presented a methodology for detecting only the few interesting sequences which were created to be traversed in that order. The proposed algorithms SC1 and SC2 use varying levels of domain knowledge, coded in terms of assumptions on the sequences of interest, but essentially capture the same notion that consecutive elements of a sequence have a constant relation between them. SC1 identifies continuity links in web pages, as well as, their positional information, and tracks sequences by traversing pages through links with the same positional information. SC2, on the other hand, operates directly on the URL list itself, identifying consecutive pages based on the URL strings. Apart from providing the algorithms to detect sequences of web documents, this chapter also highlighted some potential applications, each having great value.

- *Fair comparison* has been thoroughly discussed in Chapter 3, and is intended to tackle the possibility of a search engine being coaxed into promoting the ranks of irrelevant (or less relevant) pages.

- *Duplicate detection* is an application with immense value. At the same time, it presents great challenges, due to the very nature of how content is made available over the Web. Detecting and removing duplicates would result in huge savings in

storage requirements, and subsequently, on processing time.

- *Returning multiple pages as a single result* is another interesting approach of information retrieval that comes out of Chapter 3. This entails matching a single query across multiple documents. Present day search engines lack this capability (except for adding the anchor text into the content of the page being pointed to, in the indexing phase) and hence, present a single page as a match for a query.

Experiments conducted on the Python, WB13 and WB1 corpora demonstrated the effectiveness of SC1 and SC2 in detecting sequences, and also depicted how merging the obtained sequences affects the term frequencies and inverse document frequences for various terms present in the corpora.

We studied the page ranking problem from the surfer modeling perspective in Chapters 4 and 5. Both chapters, however, had different objectives and had described attempts at solving different problems.

Chapter 4 addressed the problem of modeling the inter-relationship between page categorization and ranking in terms of topic continuity. A new surfer model, called the Topic Continuity Model, was described. This model is based on the premise that a surfer's present and future locations are not independent of previous locations and the context that can be inferred from the history of the surfer. At any given time point, the surfer is assumed to be more likely to continue on the current topic of interest, occasionally venturing onto other topics, as in the real world. This incorporation of topic continuity is a unique feature of this methodology.

An offline algorithm developed for this purpose probabilistically estimates the surfer's current topic of interest from his/her present location, as well as, the history. In turn, the locations likely to be visited next are based on the current topic of interest, too. This resulted in a scalable and convergent iterative procedure that provides page categorizations as well as ranking simultaneously. Consequently, the joint probability matrix, whose entries denoted the probability of a surfer simultaneously being on a particular page and having a particular topic of interest, contains a wealth of information in it. The marginal

values of this matrix represent unconditional probabilities – the column totals denoting the page ranks, and the row totals denoting the interest in topics on the web. Similarly, dividing the entries of the matrix by the marginal values results in conditional probabilities – topic specific page ranks when divided by the row total, and page categorization when divided by the column total.

The novel theoretical formulation of fuzzy web surfer models in Chapter 5 addresses a different concern, namely, that of providing stability to the whole process of page rank computing. These models integrate the existing work on web surfer models and fuzzy Markov chains defined on the $\max$-$\min$ algebra. We have also provided a detailed section on the motivation behind the need for fuzzy surfer models. The definition of FuzzRank is a simple and elegant fuzzy counterpart of PageRank, which is based on the random surfer model. Experimental results confirm that FuzzRank has very similar ranking properties, and yet is more robust to noise. This robustness is a consequence of the tendency of FuzzRank to avoid a strict ranking in the absence of strong evidence to that effect. While FuzzRank may result in a large number of ties if this were the sole criterion for ranking web pages, given that several other factors, like query relevance, would be considered during the ranking process, the ability to consistently rank the pages in the presence of noise is an advantage.

The final contributory chapter (Chapter 6) is a culmination of this thesis, and involves comparing page ranking methodologies. It dealt with generalizing measures of discordance for the case when the underlying scores are known. A metric has been provided to compare score vectors directly. This metric turns out to be the Kendall distance when a parameter γ, denoting the ratio of fusing proportions, is large. Experiments of various kinds demonstrated the wide range of applications of this metric.

7.1 Scope for Further Research

In this section, we discuss the scope for future work related to the contributions made in the thesis.

Chapter 3 highlighted the applications of sequence detection. Naturally, it would be apt to have implementations that realize these applications. This includes development of

- Ranking algorithms that can consistenty compare content whether or not it is distributed across multiple pages,
- Algorithms for detecting duplicate and near duplicate content in a manner insensitive to the presentation styles, and
- Search engines capable of returning multiple pages as a single result. If individual parts of a query match different pages in a sequence, the user may be presented a sequence of web pages to be visited for satisfying his/her information need.

Also, another line of research arising out of the work on sequence detection would be to identify multi-level sequences, that is, sequences where pages are at varying depths. As an analogy, we have looked at various sections, and have identified the chapters that are made up of these sections. As a next step, some of these chapters may be identified to constitute a book.

While the simultaneous estimation of page ranking and categorization in Chapter 4 is an advantage of the topic continuity model, a theoretical proof of convergence evades us. Such a theoretical proof, if it exists, would guarantee the observations made experimentally.

Although we had presented experimental results only for page ranking and categorization, the topic continuity model can be made applicable for topic-sensitive page ranking and topic representation on the web, too. Also, since the parameter ϵ may be modified to accommodate the curiosity factor of an individual, each of the above applications may be personalized. Similarly, it has been assumed that all cross-topic transitions are equally

likely. In reality that may not be the case, mostly because some topics are more related while others are not. This indicates that ϵ itself might have to be varied for each topic-topic pair.

The topic continuity model presented in this chapter did not include the random jump factor. Recently, Nie *et al* [105], introduced another topic continuity model that incorporates the random jump factor. This factor may be incorporated in a similar manner in our model, too.

All the above enhancements to the model would further increase the computational load, and, consequently, it is imperative to have more efficient algorithms for obtaining the joint probability matrix. One may aim for a reduction in the complexity for obtaining the matrix by either trying to provide a variant of the topic continuity model that is easier to compute, or by employing some approximation algorithms.

Theoretical investigations into the properties of the fuzzy transition matrix, and fuzzy Markov chains, in general, may yield significant insights about convergence of the iterative procedure, ergodicity, sensitivity to noise, and better implementation.

There is a tremendous scope for future work based on the generalization of Kendall distance introduced in Chapter 6.

- One may study the cases where T is assumed to arise from specific distributions. While we have restricted ourselves to the Uniform distribution, which is reasonable under various assumptions, there may be particular cases where T is known to have a different distribution, say Gaussian, or perhaps, a Power-law distribution. We believe this additional knowledge may be incorporated into the formulation of the metric to obtain more appropriate distance measures.

- Obtaining the properties such as maximum and minimum of $D_\gamma(S_1, S_2)$ and $D^k_\gamma(S_1, S_2)$ for particular values of γ is another theoretically challenging work. These properties shall help in normalization of the metric and shall be useful for obtaining distance values that are independent of the number of items being scored.

- Speeding up the computation of the proposed metric is a very important aspect. Any speedup would have many implications on various existing algorithms in the literature. For example, it might result in better and faster algorithms for rank aggregation.

Integrating all the methodologies presented in this thesis would be an interesting task to do. This would mean building an intelligent web-based search system that preprocesses the available data sets, obtains model estimates from them, compares several variants of the models, extracts useful information, and ultimately creates a rich user experience.

Appendix A

Additional Results on Classification Performance from Chapter 2

In Tables 2.5 and 2.6 of Section 2.6.4, we had provided the accuracies obtained with the classification method chosen to be NaiveBayes, SVM and MaxEnt. However, for want of space, only two precision-recall plots, corresponding to NaiveBayes, were provided there. We now provide more precision-recall plots portraying the improved performance by the distribution based stemmer over other stemmers. It may be noted that the observations made and conclusions drawn in Section 2.6.4 were based on the plots available here too.

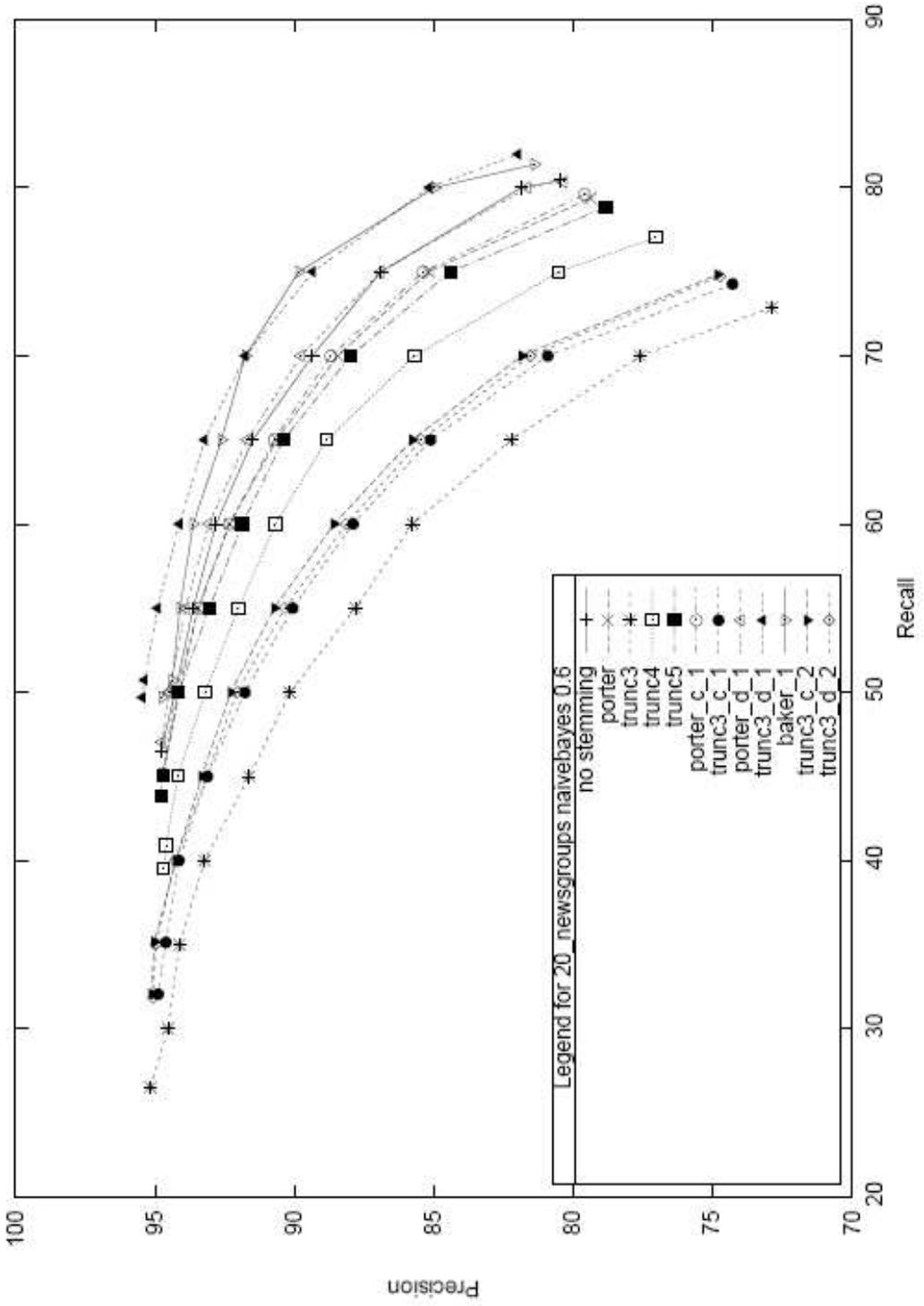


Figure A.1: Data Set: 20 newsgroups, Test Set Size: 60%, Method: NB.

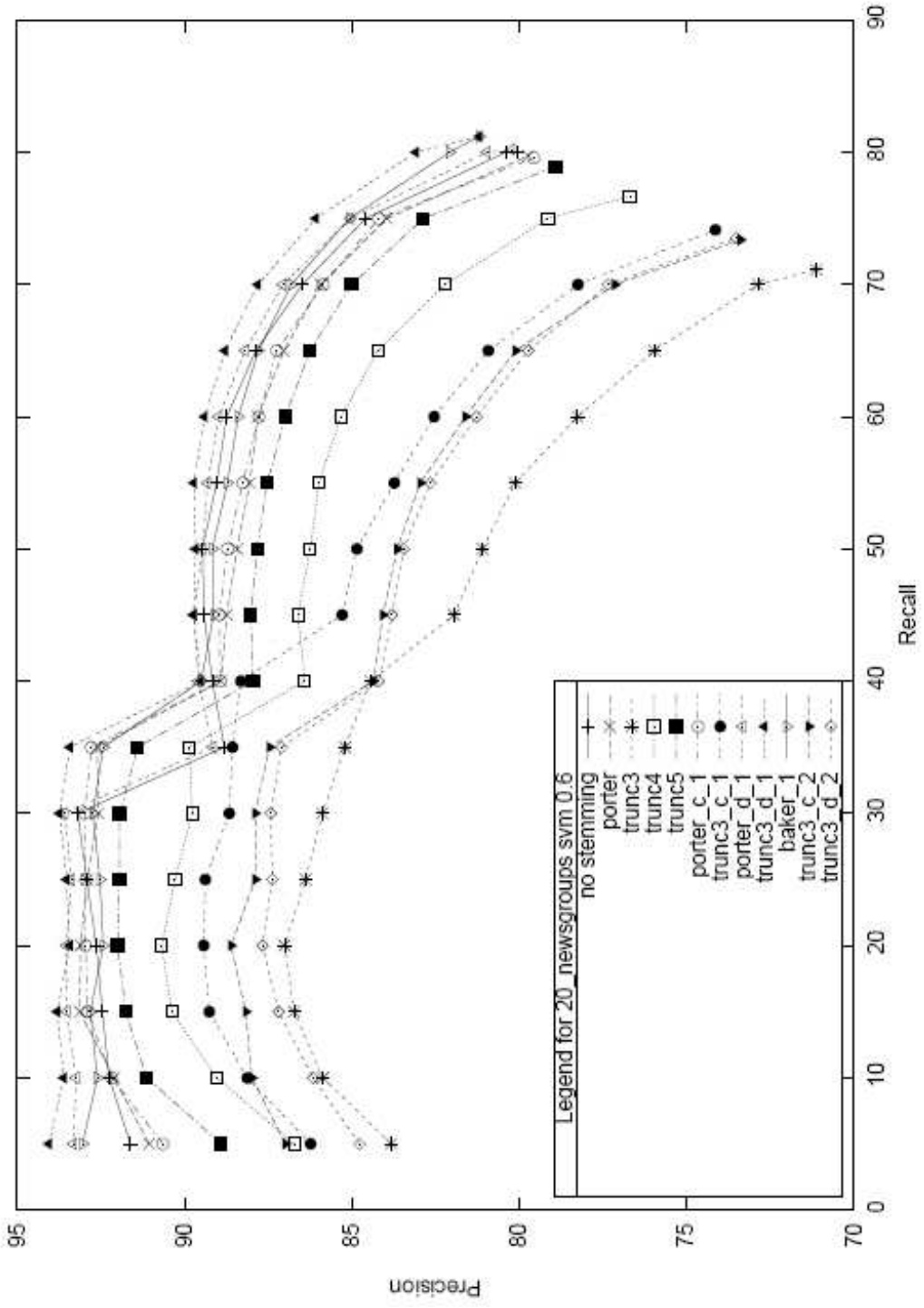


Figure A.2: Data Set: 20 newsgroups, Test Set Size: 60%, Method: SVM.

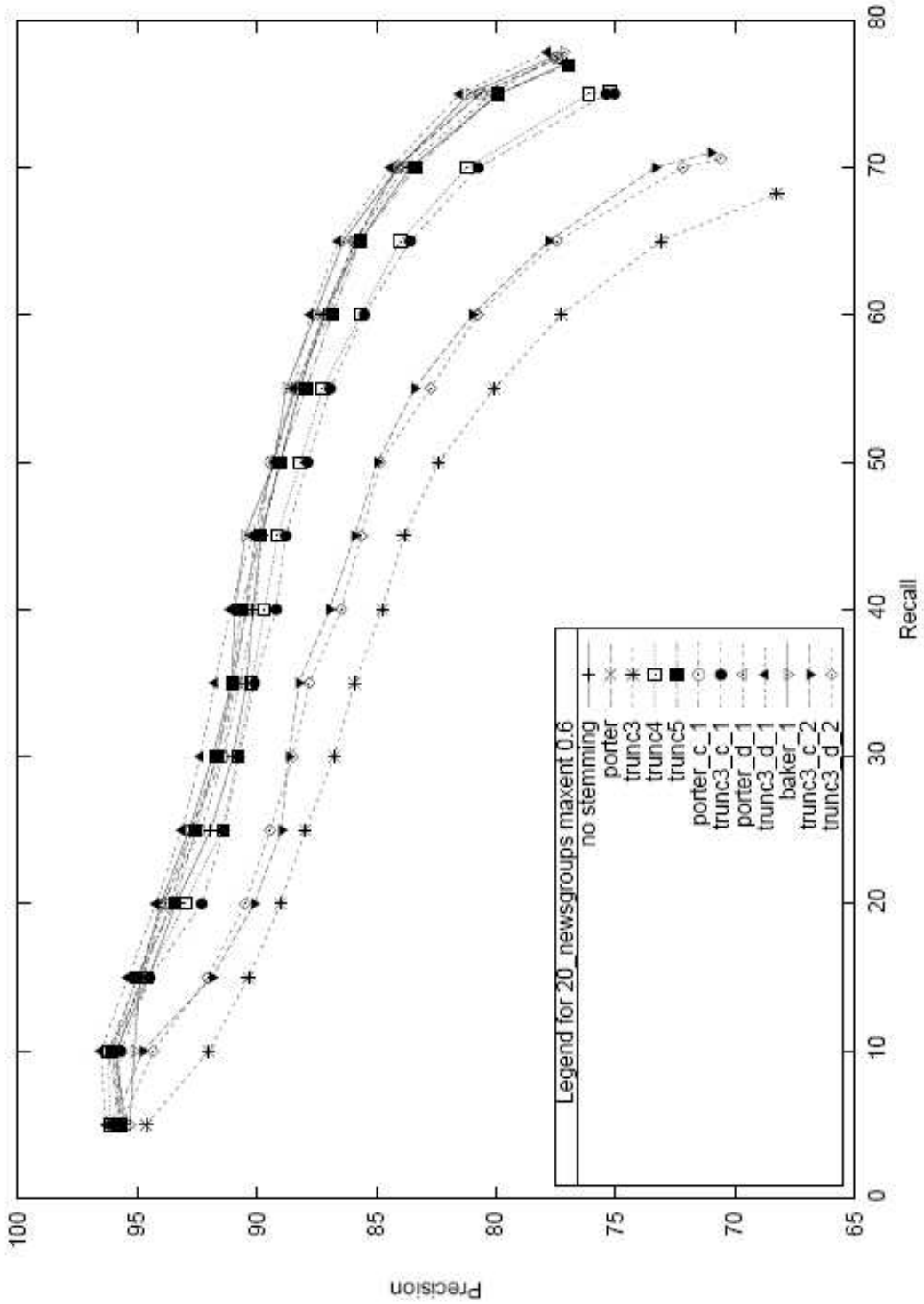


Figure A.3: Data Set: 20 newsgroups, Test Set Size: 60%, Method: MaxEnt.

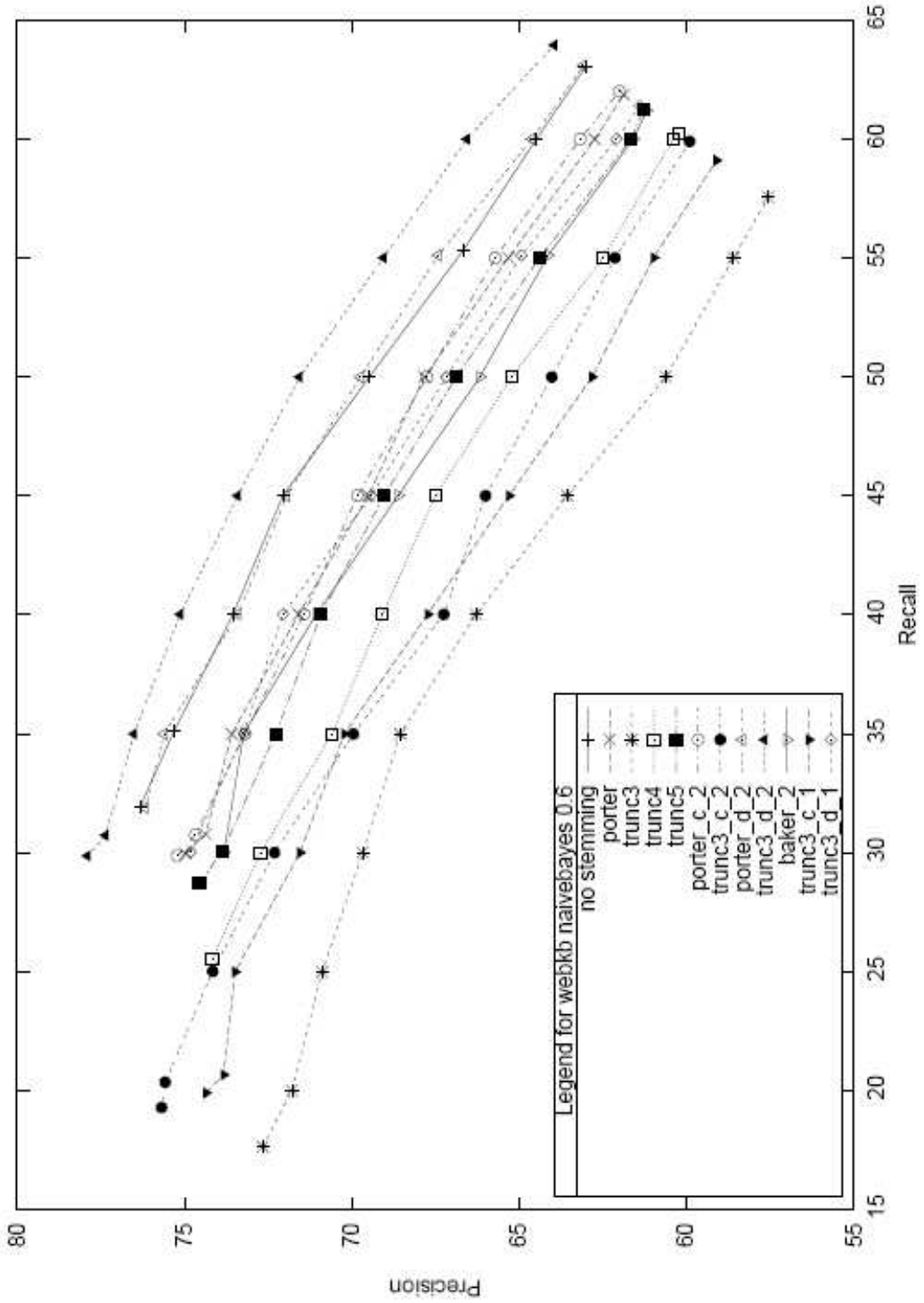


Figure A.4: Data Set: WebKB, Test Set Size: 60%, Method: NB.

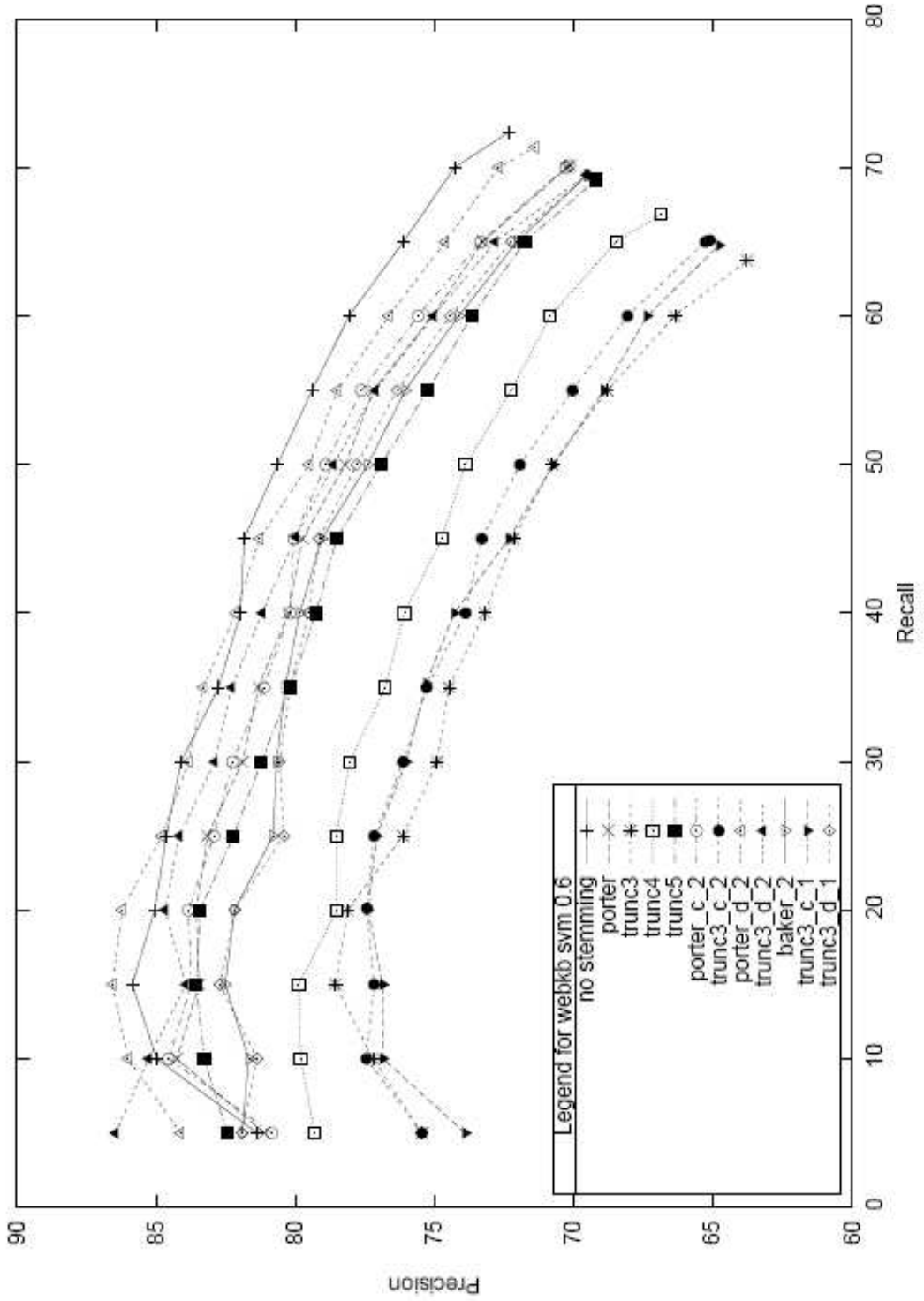


Figure A.5: Data Set: WebKB, Test Set Size: 60%, Method: SVM.

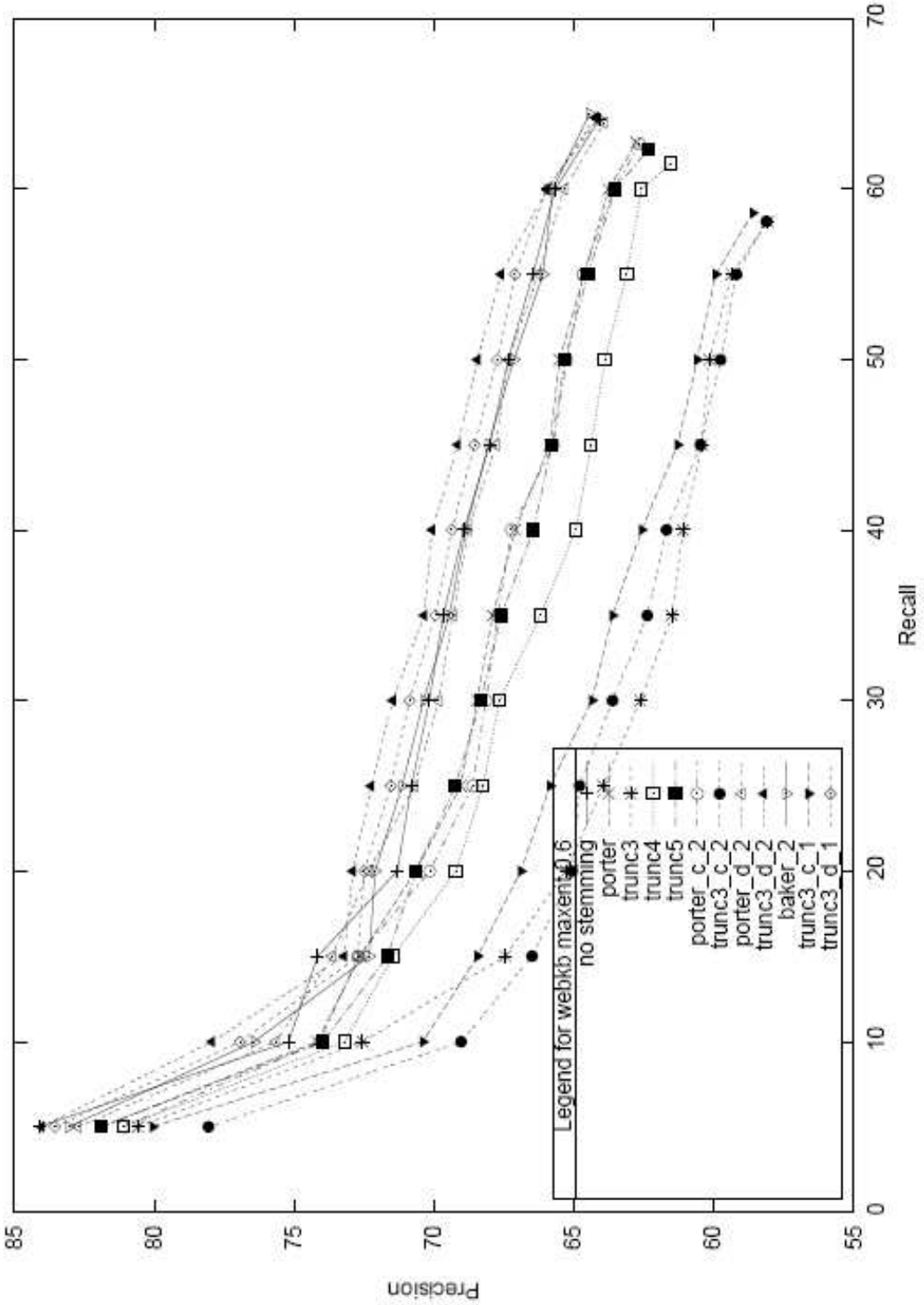


Figure A.6: Data Set: WebKB, Test Set Size: 60%, Method: MaxEnt.

Appendix B

Computation of $D^k_\gamma(S^k_1, S^k_2; i, j)$

We shall now evaluate the integrals in Eqs. 6.26–6.28. We shall denote $s_{1i} - s_{1j}$ by a in the rest of this article. (This is not an indication that a would appear as the lower limit of the integrals, though, it sometimes may). In order to evaluate the integrals, we shall repeatedly employ the following techniques:

- Split the region of integration into the positive and negative half-lines, so that $f(x)$ may be determined.
- Split the integrals to determine the upper and lower limits.
- Convert integrals of the kind $\int_a^b$ to $\int_0^b - \int_0^a$ (if $a \geq 0$) or $\int_0^b + \int_a^0$ (if $a < 0$).

Also, we shall drop the references to S_1, S_2, S^k_1 and S^k_2 as they are clear from the context.

Case 2 In order to be able to evaluate the integral in Eq. 6.26, one needs to know the answers to the following questions:

  – Whether $a \leq s_{2i} - y$: The answer to this question determines the limits of integration.

  – Whether $a < 0$: It is already known that $s_{2i} - y > 0$. So, if $a < 0$, then $\min\{a, s_{2i} - y\} = a$, and the limits of integration are known. Moreover, if

$a \geq 0$, $f(x)$ could be replaced by $1 - x$ throughout, whereas, if $a < 0$, $f(x)$ is $1 + x$ in $(a, 0)$ and $1 - x$ in $[0, \min\{\gamma(s_{2i} - y), 1\})$.

- Whether $-1 < \gamma a < 1$: If γa is the lower limit of the integration, then if $\gamma a < -1$, it is replaced by -1 and if it is greater than 1, then the integral becomes 0. Similarly, if γa is the upper limit of the integration, then if $\gamma a > 1$, it is replaced by 1 and if $\gamma a < -1$, then the integral becomes 0.
- Whether $-1 < \gamma(s_{2i} - y) < 1$. The previous observation holds again with γa replaced by $\gamma(s_{2i} - y)$.

We break up the present case into four sub-cases, whereby, the above questions may be answered better in each of the individual sub-cases.

Case 2a: $a < 0$. So, $\gamma a < 0 \leq \gamma(s_{2i} - s_{2k}) \leq \gamma(s_{2i} - y) < \gamma s_{2i}$.

$$\begin{aligned} & ED_\gamma(i, j) \\ &= \int_0^{s_{2k}} \frac{1}{s_{2k}} \left(\int_{\gamma a}^{\gamma(s_{2i}-y)} f(x)dx \right) dy \\ &= \frac{1}{s_{2k}} \int_0^{s_{2k}} \int_0^{\gamma(s_{2i}-y)} f(x)dx\, dy + \frac{1}{s_{2k}} \int_0^{s_{2k}} \int_{\gamma a}^{0} f(x)dx\, dy \\ &= I_2(s_{2i}, s_{2k}, \gamma) + I_1(-a, \gamma), \end{aligned}$$

where

$$\begin{aligned} & I_2(s_{2i}, s_{2k}, \gamma) \\ &= \frac{1}{s_{2k}} \int_0^{s_{2k}} \int_0^{\gamma(s_{2i}-y)} (1 - x)dx\, dy \\ &= \begin{cases} \frac{1}{s_{2k}} \int_0^{s_{2k}} \int_0^1 (1 - x)dx\, dy & \text{if} \quad \gamma \geq \frac{1}{s_{2i} - s_{2k}} \\ \frac{1}{s_{2k}} \left[\int_0^{s_{2i} - \frac{1}{\gamma}} \int_0^1 (1 - x)dx\, dy \right. & \\ \quad \left. + \int_{s_{2i} - \frac{1}{\gamma}}^{s_{2k}} \int_0^{\gamma(s_{2i}-y)} (1 - x)dx\, dy \right] & \text{if} \quad \frac{1}{s_{2i}} < \gamma < \frac{1}{s_{2i} - s_{2k}} \\ \frac{1}{s_{2k}} \int_0^{s_{2k}} \int_0^{\gamma(s_{2i}-y)} (1 - x)dx\, dy & \text{o.w.} \end{cases} \end{aligned}$$

$$= \begin{cases} \frac{1}{2} & \text{if} \quad \gamma \geq \frac{1}{s_{2i}-s_{2k}} \\ \frac{1}{2s_{2k}}\left(s_{2i} - \frac{1}{3\gamma} - \gamma(s_{2i}-s_{2k})^2\left(1 - \frac{\gamma(s_{2i}-s_{2k})}{3}\right)\right) & \text{if} \quad \frac{1}{s_{2i}} < \gamma < \frac{1}{s_{2i}-s_{2k}} \\ \frac{\gamma}{2}(2s_{2i} - s_{2k}) - \frac{\gamma^2}{6}(3s_{2i}^2 - 3s_{2i}s_{2k} + s_{2k}^2) & \text{o.w.} \end{cases} \tag{B.1}$$

and I_1 is as defined in Eq. 6.12.

Case 2b: $0 < a < s_{2i} - s_{2k}$. So, $a < s_{2i} - y$

$$\begin{aligned} & ED_\gamma(i,j) \\ &= \int_0^{s_{2k}} \frac{1}{s_{2k}} \left(\int_{\gamma a}^{\gamma(s_{2i}-y)} f(x)dx \right) dy \\ &= \frac{1}{s_{2k}} \int_0^{s_{2k}} \int_0^{\gamma(s_{2i}-y)} f(x)dx\, dy - \frac{1}{s_{2k}} \int_0^{s_{2k}} \int_0^{\gamma a} f(x)dx\, dy \\ &= I_2(s_{2i}, s_{2k}, \gamma) - I_1(a, \gamma), \end{aligned} \tag{B.2}$$

where I_1 and I_2 are as defined earlier.

Case 2c: $s_{2i} - s_{2k} < a < s_{2i}$ So, $\min\{a, s_{2i} - y\}$ equals a if $y \leq s_{2i} - a$, and $s_{2i} - y$ otherwise.

$$\begin{aligned} & ED_\gamma(i,j) \\ &= \frac{1}{s_{2k}} \int_0^{s_{2i}-a} \int_{\gamma a}^{\gamma(s_{2i}-y)} (1-x)dx\, dy + \frac{1}{s_{2k}} \int_{s_{2i}-a}^{s_{2k}} \int_{\gamma(s_{2i}-y)}^{\gamma a} (1-x)dx\, dy \\ &= \frac{1}{s_{2k}} \int_0^{s_{2i}-a} \left(\int_0^{\gamma(s_{2i}-y)} (1-x)dx\, dy - \int_0^{\gamma a} (1-x)dx\, dy \right) \\ &\quad + \frac{1}{s_{2k}} \int_{s_{2i}-a}^{s_{2k}} \left(\int_0^{\gamma a} (1-x)dx\, dy - \int_0^{\gamma(s_{2i}-y)} (1-x)dx\, dy \right) \\ &= \frac{2}{s_{2k}} \int_0^{s_{2i}-a} \left(\int_0^{\gamma(s_{2i}-y)} (1-x)dx\, dy - \int_0^{\gamma a} (1-x)dx\, dy \right) \\ &\quad + \frac{1}{s_{2k}} \int_0^{s_{2k}} \left(\int_0^{\gamma a} (1-x)dx\, dy - \int_0^{\gamma(s_{2i}-y)} (1-x)dx\, dy \right) \\ &= \frac{2}{s_{2k}} \int_0^{s_{2i}-a} \int_0^{\gamma(s_{2i}-y)} (1-x)dx\, dy - \frac{2(s_{2i}-a)}{s_{2k}} I_1(a, \gamma) \end{aligned}$$

$$
\begin{aligned}
& + I_1(a,\gamma) - I_1(s_{2i}, s_{2k}, \gamma) \\
= \; & \frac{2(s_{2i}-a)}{s_{2k}} \left(I_2(s_{2i}, s_{2i}-a, \gamma) - I_1(a,\gamma)\right) + I_1(a,\gamma) - I_2(s_{2i}, s_{2k}, \gamma)
\end{aligned} \tag{B.3}
$$

Case 2d: $a > s_{2i}$ Thus, $\max\{a, s_{2i} - y\}$ equals a, and we have

$$
\begin{aligned}
& ED_\gamma(i,j) \\
= \; & \int_0^{s_{2k}} \frac{1}{s_{2k}} \left(\int_{\gamma(s_{2i}-y)}^{\gamma a} (1-x)dx \right) dy \\
= \; & I_1(a,\gamma) - I_2(s_{2i}, s_{2k}, \gamma),
\end{aligned}
$$

Case 3: The integral in Eq. 6.27 is relatively easier to evaluate. Letting $y = s_{1j}$ and $z = s_{2i}$, and noting that $z - s_{2j} \leq 0 \leq s_{1i} - y$, the expected value of the degree of discordance may be computed as

$$
\begin{aligned}
& D_\gamma^k(S_1^k, S_2^k; i, j) = ED_\gamma(S_1, S_2; i, j) \\
= \; & \int_0^{s_{1k}} \int_0^{s_{2k}} \frac{1}{s_{1k}s_{2k}} \left(\int_{\gamma(z-s_{2j})}^{\gamma(s_{1i}-y)} f(x)dx \right) dz \; dy \\
= \; & \int_0^{s_{1k}} \int_0^{s_{2k}} \frac{1}{s_{1k}s_{2k}} \left(\int_0^{\gamma(s_{1i}-y)} (1-x)dx + \int_{\gamma(z-s_{2j})}^{0} (1+x)dx \right) dz \; dy \\
= \; & \frac{1}{s_{1k}} \int_0^{s_{1k}} \int_0^{\gamma(s_{1i}-y)} (1-x)dx \; dy + \frac{1}{s_{2k}} \int_0^{s_{2k}} \int_{\gamma(z-s_{2j})}^{0} (1+x)dx \; dz \\
= \; & I_2(s_{1i}, s_{1k}, \gamma) + I_2(s_{2i}, s_{2k}, \gamma)
\end{aligned} \tag{B.4}
$$

Case 4: To evaluate the expectation in Eq. 6.28, we shall split this case into two sub-cases. Let us denote s_{2i} and s_{2j} by y and z, respectively, and without loss of generality, assume that $a \geq 0$.

Case 4a: $a < s_{2k}$. The intervals of integration are split into regions where $a \leq y - z$ and $a > y - z$, respectively. Thus, Eq. 6.28 may be rewritten as

$$
\begin{aligned}
& ED_\gamma(i,j) \\
= \; & \frac{1}{s_{2k}^2} \int_0^{s_{2k}} \int_0^{s_{2k}} \int_{\min\{\gamma a, \gamma(y-z)\}}^{\max\{\gamma a, \gamma(y-z)\}} f(x)dx \; dz \; dy
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{s_{2k}^2}\int_0^a\int_0^{s_{2k}}\int_{\gamma(y-z)}^{\gamma a} f(x)dx\,dz\,dy \\
&\quad +\frac{1}{s_{2k}^2}\int_a^{s_{2k}}\left(\int_0^{y-a}\int_{\gamma a}^{\gamma(y-z)} f(x)dx\,dz\ +\int_{y-a}^{s_{2k}}\int_{\gamma(y-z)}^{\gamma a} f(x)dx\,dz\right)dy \\
&= \frac{1}{s_{2k}^2}\int_0^a\left(\int_0^y\int_{\gamma(y-z)}^{\gamma a}(1-x)dx\,dz+\int_y^{s_{2k}}\int_{\gamma(y-z)}^{\gamma a} f(x)dx\,dz\right)dy \\
&\quad +\frac{1}{s_{2k}^2}\int_a^{s_{2k}}\left(\int_0^{y-a}\int_{\gamma a}^{\gamma(y-z)}(1-x)dx\,dz+\int_{y-a}^{y}\int_{\gamma(y-z)}^{\gamma a}(1-x)dx\,dz\right. \\
&\quad \left.+\int_y^{s_{2k}}\int_{\gamma(y-z)}^{\gamma a} f(x)dx\,dz\right)dy \\
&= \frac{1}{s_{2k}^2}\int_0^a\left(\int_0^y\left(\int_0^{\gamma a}(1-x)dx\ -\int_0^{\gamma(y-z)}(1-x)dx\right)dz\right. \\
&\quad \left.+\int_y^{s_{2k}}\left(\int_0^{\gamma a}(1-x)dx+\int_{\gamma(y-z)}^{0}(1+x)dx\right)dz\right)dy \\
&\quad +\frac{1}{s_{2k}^2}\int_a^{s_{2k}}\left(\int_0^{y-a}\left(\int_0^{\gamma(y-z)}(1-x)dx-\int_0^{\gamma a}(1-x)dx\right)dz\right. \\
&\quad \left.+\int_{y-a}^{y}\left(\int_0^{\gamma a}(1-x)dx-\int_0^{\gamma(y-z)}(1-x)dx\right)dz\right. \\
&\quad \left.+\int_y^{s_{2k}}\left(\int_0^{\gamma a}(1-x)dx+\int_{\gamma(y-z)}^{0}(1+x)dx\right)dz\right)dy \\
&= \frac{1}{s_{2k}^2}\left(\int_0^{s_{2k}}\int_0^{s_{2k}}\int_0^{\gamma a}(1-x)dx\,dz\,dy-2\int_a^{s_{2k}}\int_0^{y-a}\int_0^{\gamma a}(1-x)dx\,dz\,dy\right) \\
&\quad +\frac{1}{s_{2k}^2}\int_0^{s_{2k}}\int_y^{s_{2k}}\int_{\gamma(y-z)}^{0}(1+x)dx\,dz\,dy \\
&\quad +\frac{1}{s_{2k}^2}\left(\int_a^{s_{2k}}\int_0^{y-a}\int_0^{\gamma(y-z)}(1-x)dx\,dz\,dy\right. \\
&\quad \left.-\int_0^a\int_0^y\int_0^{\gamma(y-z)}(1-x)dx\,dz\,dy-\int_a^{s_{2k}}\int_{y-a}^{y}\int_0^{\gamma(y-z)}(1-x)dx\,dz\,dy\right) \\
&= I_1(a,\gamma)-\frac{(s_{2k}-a)^2}{s_{2k}^2}I_1(a,\gamma) \\
&\quad +\frac{1}{s_{2k}^2}\int_0^{s_{2k}}\int_0^{s_{2k}-y}\int_0^{\gamma u}(1-x)dx\,du\,dy \\
&\quad +\frac{1}{s_{2k}^2}\left(\int_0^{s_{2k}}\int_0^y\int_0^{\gamma u}(1-x)dx\,du\,dy-2\int_0^a\int_0^y\int_0^{\gamma u}(1-x)dx\,du\,dy\right.
\end{aligned}
$$

$$
\begin{aligned}
& -2\int_a^{s_{2k}}\int_{y-a}^{y}\int_0^{\gamma u}(1-x)dx\,du\,dy\Big) \\
= \;& I_1(a,\gamma) - \frac{(s_{2k}-a)^2}{s_{2k}^2} I_1(a,\gamma) + 2I_3(s_{2k}, s_{2k}, \gamma) - 2I_3(a, s_{2k}, \gamma) \\
& -2I_4(a, s_{2k}, \gamma), \qquad \text{(B.5)}
\end{aligned}
$$

where,

$$
\begin{aligned}
I_3(a, s_{2k}, \gamma) &= \frac{1}{s_{2k}^2}\int_0^a\int_0^y\int_0^{\gamma u}(1-x)dx\,du\,dy \\
&= \begin{cases}
\frac{1}{s_{2k}^2}\int_0^a\int_0^y\left(\gamma u - \frac{\gamma^2u^2}{2}\right)du\,dy & \text{if} \quad \gamma a \le 1 \\
\frac{1}{s_{2k}^2}\int_0^{\frac{1}{\gamma}}\int_0^y\left(\gamma u - \frac{\gamma^2u^2}{2}\right)du\,dy & \\
+\frac{1}{s_{2k}^2}\int_{\frac{1}{\gamma}}^a\int_0^{\frac{1}{\gamma}}\left(\gamma u - \frac{\gamma^2u^2}{2}\right)du\,dy & \text{o.w.} \\
+\frac{1}{s_{2k}^2}\int_{\frac{1}{\gamma}}^a\int_{\frac{1}{\gamma}}^y\frac{1}{2}du\,dy &
\end{cases} \\
&= \begin{cases}
\frac{\gamma a^3}{6s_{2k}^2}\left(1-\frac{\gamma a}{4}\right) & \text{if} \quad \gamma a \le 1 \\
\frac{1}{4s_{2k}^2}\left(\frac{1}{2\gamma^2} + \left(a-\frac{1}{\gamma}\right)\left(a+\frac{1}{3\gamma}\right)\right) & \text{o.w.}
\end{cases}
\end{aligned}
$$

and

$$
\begin{aligned}
I_4(a, s_{2k}, \gamma) &= \frac{1}{s_{2k}^2}\int_a^{s_{2k}}\int_0^a\int_0^{\gamma u}(1-x)dx\,du\,dy \\
&= \begin{cases}
\frac{s_{2k}-a}{s_{2k}^2}\int_0^a\left(\gamma u - \frac{\gamma^2u^2}{2}\right)du\,dy & \text{if} \quad \gamma a \le 1 \\
\frac{s_{2k}-a}{s_{2k}^2}\left(\int_0^{\frac{1}{\gamma}}\left(\gamma u - \frac{\gamma^2u^2}{2}\right)du\,dy + \int_{\frac{1}{\gamma}}^a\frac{1}{2}du\,dy\right) & \text{o.w.}
\end{cases} \\
&= \begin{cases}
\frac{s_{2k}-a}{s_{2k}^2}\frac{\gamma a^2}{2}\left(1-\frac{\gamma a}{3}\right) & \text{if} \quad \gamma a \le 1 \\
\frac{s_{2k}-a}{s_{2k}^2}\left(a-\frac{2}{3\gamma}\right) & \text{o.w.}
\end{cases}
\end{aligned}
$$

Case 4b: $a \ge s_{2k}$.

$$
\begin{aligned}
& ED_\gamma(i,j) \\
&= \frac{1}{s_{2k}^2}\int_0^{s_{2k}}\int_0^{s_{2k}}\int_{\gamma(y-z)}^{\gamma a} f(x)dx\,dz\,dy \\
&= \frac{1}{s_{2k}^2}\int_0^{s_{2k}}\left(\int_0^y\int_{\gamma(y-z)}^{\gamma a}(1-x)dx\,dz + \int_y^{s_{2k}}\int_{\gamma(y-z)}^{\gamma a} f(x)dx\,dz\right)dy
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{s_{2k}^2}\int_0^{s_{2k}}\int_0^{y}\left(\int_0^{\gamma a}(1-x)dx - \int_0^{\gamma(y-z)}(1-x)dx\right)dzdy \\
&\quad + \frac{1}{s_{2k}^2}\int_0^{s_{2k}}\int_y^{s_{2k}}\left(\int_0^{\gamma a}(1-x)dx + \int_{\gamma(y-z)}^{0}(1+x)dx\right)dzdy \\
&= \frac{1}{s_{2k}^2}\int_0^{s_{2k}}\int_0^{s_{2k}}\int_0^{\gamma a}(1-x)dx\,dzdy \\
&\quad - \frac{1}{s_{2k}^2}\int_0^{s_{2k}}\int_0^{y}\int_0^{\gamma(y-z)}(1-x)dx + \frac{1}{s_{2k}^2}\int_0^{s_{2k}}\int_y^{s_{2k}}\int_{\gamma(y-z)}^{0}(1+x)dx\,dz\,dy \\
&= I_1(a,\gamma) - I_3(s_{2k}, s_{2k}, \gamma) + I_3(s_{2k}, s_{2k}, \gamma) = I_1(a,\gamma) \qquad \text{(B.6)}
\end{aligned}
$$

Bibliography

[1] "20 news groups data set," http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.

[2] M. Ángeles Serrano, A. Maguitman, n. Marián Bogu S. Fortunato, and A. Vespignani, "Decoding the structure of the WWW: A comparative analysis of web crawls," *ACM Transactions on the Web*, vol. 1, no. 2, p. 10, 2007.

[3] D. Arotaritei and S. Mitra, "Web mining: a survey in the fuzzy framework," *Fuzzy Sets and Systems*, vol. 148, pp. 5–19, 2004.

[4] K. E. Avrachenkov and E. Sanchez, "Fuzzy markov chains and decision-making," *Fuzzy Optimization and Decision Making*, vol. 1, no. 2, pp. 143–159, 2002.

[5] M. Bacchin, N. Ferro, and M. Melucci, "A probabilistic model for stemmer generation," *Information Processing and Management*, vol. 41, no. 1, pp. 121–137, 2005.

[6] L. D. Baker and A. K. McCallum, "Distributional clustering of words for text classification," in *Proceedings of Twenty First ACM International Conference on Research and Development in Information Retrieval*, Melbourne, Australia, pp. 96–103, 1998.

[7] P. Baldi, P. Frasconi, and P. Smyth, *Modeling the Internet and the Web : Probabilistic Methods and Algorithms*. John Wiley and Sons, 2003.

[8] S. Bandyopadhyay and S. K. Pal, *Classification and Learning using Genetic Algorithms*. Heidelberg: Springer, 2007.

[9] J. Bar-Ilan, M. Mat-Hassan, and M. Levene, "Methods for comparing rankings of search engine results," *Computer Networks*, vol. 50, no. 10, pp. 1448–1463, 2006.

[10] G. P. Basharin, A. N. Langville, and V. A. Naumov, "The life and work of A. A. Markov," *Linear Algebra and its Applications*, vol. 386, pp. 3–26, 2004.

[11] P. Berkhin, "A survey on PageRank computing," *Internet Mathematics*, vol. 2, no. 1, pp. 73–120, 2005.

[12] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret, "The World-Wide Web," *Communications of the ACM*, vol. 37, no. 8, pp. 76–82, 1994.

[13] N. L. Bhamidipati and S. K. Pal, "Comparing rank-inducing scoring systems," in *Proceedings of the Eighteenth International Conference on Pattern Recognition (ICPR06)*, Hong Kong, pp. 300–303, 2006.

[14] N. L. Bhamidipati and S. K. Pal, "Stemming via distribution-based segregation for classification and retrieval," *IEEE Transactions on Systems, Man and Cybernetics B*, vol. 37, no. 2, pp. 350–360, 2007.

[15] N. L. Bhamidipati and S. K. Pal, *Web Intelligence Meets Brain Informatics*. Berlin: Springer Verlag, 2007, ch. Fuzzy Web Surfer Models: Theory and Experiments, pp. 324–340.

[16] N. L. Bhamidipati and S. K. Pal, "Comparing scores intended for ranking," *IEEE Transactions on Knowledge and Data Engineering*, Revised and Submitted.

[17] K. Bharat and G. A. Mihaila, "When experts agree: using non-affiliated experts to rank popular topics," in *Proceedings of the Tenth International World Wide Web Conference*, Hong Kong, pp. 597–602, 2001.

[18] Z. Bi, C. Faloutsos, and F. Korn, "The DGX distribution for mining massive, skewed data," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, pp. 17–26, 2001.

[19] P. Boldi and S. Vigna, "The WebGraph framework I: Compression techniques," in *Proceedings of the Thirteenth International World Wide Web Conference*, NY, USA, pp. 595 – 602, 2004.

[20] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas, "Link analysis ranking: algorithms, theory, and experiments," *ACM Transactions on Internet Technology*, vol. 5, no. 1, pp. 231–297, 2005.

[21] R. P. Brent, "An improved Monte Carlo factorization algorithm," *BIT*, vol. 20, pp. 176–184, 1980.

[22] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Proceedings of the Seventh International World Wide Web Conference*, Brisbane, Australia, pp. 107–117, 1998.

[23] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the Web," in *Proceedings of the Ninth International World Wide Web Conference*, Amsterdam, Netherlands, pp. 309–320, 2000.

[24] J. J. Buckley, *Fuzzy probabilities and fuzzy sets for web planning*. Springer, 2004.

[25] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A recursive model for graph mining," in *Proceedings of SIAM Data Mining*, Lake Buena Vista, FL, USA, 2004. [Online]. Available: http://www.siam.org/meetings/sdm04/proceedings/sdm04_043.pdf

[26] S. Chakrabarti, *Mining the Web: Discovering knowledge from hypertext data*. Morgan Kaufman, 2002.

[27] S. Chakrabarti, B. E. Dom, D. Gibson, J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Mining the link structure of the world wide web," *IEEE Computer*, vol. 32, no. 8, pp. 60–67, 1999.

[28] S. Chakrabarti, M. Joshi, K. Punera, and D. Pennock, "The structure of broad topics on the web," in *Proceedings of the Eleventh International World Wide Web Conference*, Honolulu, HI, USA, pp. 251–262, 2002.

[29] S. Chakrabarti, M. M. Joshi, and V. B. Tawde, "Enhanced topic distillation using text, markup tags and hyperlinks," in *Proceedings of the Twenty Fourth Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, New Orleans, LA, USA, pp. 208–216, 2001.

[30] A. M. Cohen, J. Yang, and W. R. Hersh, "A comparison of techniques for classification and ad hoc retrieval of biomedical documents," in *Proceedings of the Fourteenth Annual Text REtrieval Conference*, Gaithersburg, MD, USA, pp. 628–636, 2005.

[31] D. Cohn and H. Chang, "Learning to probabilistically identify authoritative documents," in *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford, CA, USA, pp. 167–174, 2000.

[32] W. J. Conover, *Practical Nonparametric Statistics*, 3rd ed. John Wiley and Sons, 1999.

[33] R. Cooley, B. Mobasher, and J. Srivastava, "Web Mining: Information and pattern discovery on the World Wide Web," in *Proceedings of the Ninth IEEE International Conference on Tools with Artificial Intelligence*, Newport Beach, CA, pp. 558–567, 1997.

[34] F. Crestani, "Combination of similarity measures for effective spoken document retrieval," *Journal of Information Science*, vol. 29, no. 2, pp. 87–96, 2003.

[35] F. Crestani and G. Pasi, *Neuro-Fuzzy Techniques for Intelligent Information Systems*. Heidelberg, Germany: Physica Verlag, 1999, ch. Soft Information Retrieval: Applications of Fuzzy Set Theory and Neural Networks, pp. 287–315.

[36] B. D. Davison, "Unifying text and link analysis," in *IJCAI-03 Workshop on Text-Mining & Link-Analysis (TextLink)*, Acapulco, Mexico, 2003. [Online]. Available: http://www.cse.lehigh.edu/~brian/pubs/2003/textlink/short.pdf

[37] B. D. Davison, "Recognizing nepotistic links on the web," in *Artificial Intelligence for Web Search*, Austin, TX, USA, pp. 23–28, 2000.

[38] J. L. Dawson, "Suffix removal for word conflation," *Bulletin of the Association for Literary and Linguistic Computing*, vol. 2, no. 3, pp. 33–46, 1974.

[39] R. S. de Madariaga, J. R. F. del Castillo, and J. R. Hilera, "A generalization of the method for evaluation of stemming algorithms based on error counting," in *Proceedings of the Twelfth International Conference on String Processing and Information Retrieval*, Buenos Aires, Argentina, pp. 228–233, 2005.

[40] B. Debroy and L. Bhandari, "Economic freedom for the states of India," Rajiv Gandhi Institute for Contemporary Studies, New Delhi, India, Tech. Rep., 2005.

[41] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*. New Jersey: Prentice Hall, 1982.

[42] M. Diligenti, M. Gori, and M. Maggini, "A unified probabilistic framework for web page scoring systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 4–16, 2004.

[43] S. Dominich, "PageRank: Quantitative model of interaction information retrieval," in *Proceedings of the Twelfth International World Wide Web Conference*, Budapest, Hungary, pp. 13–18, 2003.

[44] K. M. Donald and A. F. Smeaton, "A comparison of score, rank and probability-based fusion methods for video shot retrieval," in *Proceedings of International Conference on Image and Video Retrieval*, Urbana, IL, USA, pp. 61–70, 2003.

[45] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley-Interscience, 2000.

[46] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proceedings of the Tenth International World Wide Web Conference*, Hong Kong, pp. 613–622, 2001.

[47] P. Erdos and A. Renyi, "On random graphs I," *Publicationes Mathematicae*, pp. 290–297, 1959.

[48] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top k lists," *Siam Journal of Discrete Mathematics*, vol. 17, no. 1, pp. 134–160, 2003.

[49] U. Fayyad and R. Uthurusamy, "Data mining and knowledge discovery in databases," *Communications of the ACM*, vol. 39, no. 11, pp. 24–27, 1996.

[50] W. B. Frakes and C. J. Fox, "Strength and similarity of affix removal stemming algorithms," *ACM SIGIR Forum*, vol. 37, no. 1, pp. 26–30, 2003.

[51] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. San Diego: Academic Press, 1990.

[52] "Google Inc." http://www.google.com/.

[53] "Google search index size," http://www.google.com/help/indexsize.html.

[54] T. Gustad and G. Bouma, "Accurate stemming of Dutch for text classification," *Language and Computers*, vol. 45, no. 1, pp. 104–117, 2002.

[55] V. Ha and P. Haddawy, "Similarity of personal preferences: theoretical foundations and empirical analysis," *Artificial Intelligence*, vol. 146, no. 2, pp. 149–173, 2003.

[56] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. San Mateo, CA: Morgan Kaufmann, 2000.

[57] F. Harary, *Graph Theory*. Narosa Publications, 1988.

[58] D. Harman, "Overview of the Third Text REtrieval Conference," in *Proceedings of the Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD, USA, pp. 1–20, 1995.

[59] D. Harman, "How effective is suffixing?" *Journal of the American Society for Information Science*, vol. 42, no. 1, pp. 7–15, 1991.

[60] T. H. Haveliwala, "Efficient computation of pagerank," Stanford University, Tech. Rep., 1999.

[61] T. H. Haveliwala, "Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 784–796, 2003.

[62] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proceedings of the Eleventh International World Wide Web Conference*, Honolulu, HI, USA, pp. 517–526, 2002.

[63] D. Hawking, N. Craswell, P. Bailey, and K. Griffiths, "Measuring search engine quality," *Information Retrieval*, vol. 4, no. 1, pp. 33–59, 2001.

[64] E. Herrera-Viedma and G. Pasi, "Soft approaches to information retrieval and information access on the web," *Journal of the American Society for Information Science and Technology*, vol. 57, no. 4, pp. 511–514, 2006.

[65] J. Hirai, S. Raghavan, A. Paepcke, and H. Garcia-Molina, "WebBase : A repository of web pages," in *Proceedings of the Ninth International World Wide Web Conference*, Amsterdam, Netherlands, pp. 277–293, 2000.

[66] K. Hirota and W. Pedrycz, "Fuzzy computing for data mining," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1575–1600, 1999.

[67] P. G. Hoel, S. C. Port, and C. J. Stone, *Introduction to Stochastic Processes*. Waveland Press, Incorporated, 1972.

[68] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," in *Proceedings of the Tenth European Conference on Machine Learning*, Chemnitz, Germany, pp. 137–142, 1998.

[69] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Discrete Multivariate Distributions*. Wiley-Interscience, 1997.

[70] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, "Extrapolation methods for accelerating pagerank computations," in *Proceedings of the Twelfth International World Wide Web Conference*, Budapest, Hungary, pp. 261–270, 2003.

[71] M. Kantrowitz, B. Mohit, and V. Mittal, "Stemming and its effects on TFIDF ranking," in *Proceedings of the Twenty Third Annual SIGIR conference on Research and Development in Information Retrieval*, Athens, Greece, pp. 357–359, 2000.

[72] S.-W. Kim and B. J. Oommen, "Enhancing prototype reduction schemes with recursion: A method applicable for "large" data sets," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 34, no. 3, pp. 1384–1397, 2004.

[73] J. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[74] W. R. Knight, "A computer method for calculating Kendall's tau with ungrouped data," *Journal of the American Statistical Association*, vol. 61, no. 314, pp. 436–439, 1966.

[75] A. Kolakowska and W. Malina, "Fisher sequential classifiers," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 35, no. 5, pp. 988–998, 2005.

[76] R. Kosala and H. Blockeel, "Web mining research: a survey," *SIGKDD Explorations Newsletter*, vol. 2, no. 1, pp. 1–15, 2000.

[77] W. Kraaij and R. Pohlmann, "Viewing stemming as recall enhancement," in *Proceedings of the Seventeenth ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, pp. 40–48, 1996.

[78] R. Krovetz, "Viewing morphology as an inference process," in *Proceedings of the Sixteenth ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, PA, USA, pp. 191–202, 1993.

[79] R. Kruse, R. Buck-Emden, and R. Cordes, "Processor power considerations - an application of fuzzy markov chains," *Fuzzy Sets and Systems*, vol. 21, pp. 289–299, 1987.

[80] K. Kukich, "Technique for automatically correcting words in text," *ACM Computing Surveys*, vol. 24, no. 4, pp. 377–439, 1992.

[81] S. Kullback, *Information theory and statistics*. New York: Wiley, 1959.

[82] A. N. Langville and C. D. Meyer, "A survey of eigenvector methods for web information retrieval," *SIAM Review*, vol. 47, no. 1, pp. 135–161, 2005.

[83] J. H. Lee, "Analyses of multiple evidence combination," in *Proceedings of the Twentieth Annual International ACM SIGIR Conference in Research and Development in Information Retrieval*, Philadelphia, PA, USA, pp. 267–276, 1995.

[84] J. H. Lee, "Combining multiple evidence from different properties of weighting schemes," in *Proceedings of the Eighteenth Annual International ACM SIGIR Conference in Research and Development in Information Retrieval*, Seattle, WA, USA, pp. 180–188, 1995.

[85] E. L. Lehmann, *Testing Statistical Hypotheses*. New York: Springer-Verlag, 1997.

[86] R. Lempel and S. Moran, "SALSA: The stochastic approach for link-structure analysis," *ACM Transactions on Information Systems*, vol. 19, no. 2, pp. 131–160, 2001.

[87] K. Lerman, L. Getoor, S. Minton, and C. Knoblock, "Using the structure of web sites for automatic segmentation of tables," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, Paris, France, pp. 119–130, 2004.

[88] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval." in *Proceedings of the Tenth European Conference on Machine Learning*, Chemnitz, Germany, pp. 4–15, 1998.

[89] D. D. Lewis, "Feature selection and feature extraction for text categorization," in *Proceedings of the workshop on Speech and Natural Language*, NY, USA, pp. 212–217, 1992.

[90] Y. Li, B. Murphy, and N. Zhong, "Mining interesting topics for web information gathering and web personalization," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, Compiégne, France, pp. 305–308, 2005.

[91] B. Liu, *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Berlin: Springer, 2007.

[92] J. Liu, "Web intelligence (WI): What makes wisdom web?" in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 1596–1601, 2003.

[93] V. Loia, W. Pedrycz, S. Senatore, and M. I. Sessa, "Web navigation support by means of proximity-driven assistant agents," *Journal of the American Society for Information Science and Technology*, vol. 57, no. 4, pp. 515–527, 2006.

[94] G. Loizou and P. Thanisch, "Enumerating the cycles of a digraph: A new preprocessing strategy," *Information Sciences*, vol. 27, pp. 163–182, 1982.

[95] J. B. Lovins, "Development of a stemming algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, pp. 22–31, 1968.

[96] A. K. McCallum, "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering," 1996, http://www.cs.cmu.edu/∼mccallum/bow.

[97] M. Mitzenmacher, "A brief history of generative models for power law and lognormal distributions," *Internet Mathematics*, vol. 1, no. 2, pp. 226–251, 2004.

[98] M. Montague, "Metasearch: Data fusion for document retrieval," Ph.D. dissertation, Dartmouth College, Hanover, NH. USA, 2002.

[99] M. Montague and J. A. Aslam, "Relevance score normalization for metasearch," in *Proceedings of the Tenth international conference on Information and knowledge management*, Atlanta, GA, USA, pp. 427–433, 2001.

[100] S. A. Mounir, N. Goharian, M. Mahoney, A. Salem, and O. Frieder, "Fusion of information retrieval engines (FIRE)," in *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications,* Las Vegas, NV, USA, 1998. [Online]. Available: http://www.ir.iit.edu/publications/downloads/cse6000.pdf

[101] B. L. Narayan, C. A. Murthy, and S. K. Pal, "Topic continuity for web document categorization and ranking," in *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, pp. 310–315, 2003.

[102] B. L. Narayan and S. K. Pal, "Detecting sequences and cycles of web pages," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, Compiégne, France, pp. 80–86, 2005.

[103] B. L. Narayan and S. K. Pal, "Distribution based stemmer refinement," in *Proceedings of the First International Conference on Pattern Recognition and Machine Intelligence*, Kolkata, India, pp. 672–677, 2005.

[104] B. L. Narayan and S. K. Pal, "A fuzzy web surfer model," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, Compiégne, France, pp. 120–123, 2005.

[105] L. Nie, B. D. Davison, and X. Qi, "Topical link analysis for web search," in *Proceedings of the Twenty Ninth Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, Seattle, WA, pp. 91–98, 2006.

[106] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," in *Proceedings of IJCAI-99 Workshop on Machine Learning for Information Filtering*, Stockholm, Sweden, pp. 61–67, 1999.

[107] G. Nivasch, "Cycle detection using a stack," *Information Processing Letters*, vol. 90, no. 3, pp. 135–140, 2004.

[108] R. Nuray and F. Can, "Automatic ranking of information retrieval systems using data fusion," *Information Processing & Management*, vol. 42, no. 3, pp. 595–614, 2006.

[109] "Open directory project," http://dmoz.org/about.html.

[110] C. D. Paice, "Another stemmer," *SIGIR Forum*, vol. 24, no. 3, pp. 56–61, 1990.

[111] C. D. Paice, "A method for the evaluation of stemming algorithms based on error counting," *Journal of the American Society for Information Science*, vol. 47, no. 8, pp. 632–649, 1996.

[112] S. K. Pal, V. Talwar, and P. Mitra, "Web mining in soft computing framework: Relevance, state of the art and future directions," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1163–1177, 2002.

[113] S. K. Pal and D. D. Majumder, *Fuzzy mathematical approach to pattern recognition*. New York: John Wiley, 1986.

[114] S. K. Pal and P. Mitra, *Pattern Recognition Algorithms for Data Mining*. Boca Raton, FL: CRC Press, 2004.

[115] S. K. Pal, B. L. Narayan, and S. Dutta, "A web surfer model incorporating topic continuity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 5, pp. 726–729, 2005.

[116] S. K. Pal and A. Pal, Eds., *Pattern Recognition: From Classical to Modern Approaches*. Singapore: World Scientific, 2001.

[117] W. Pedrycz and F. Gomide, *An Introduction to Fuzzy Sets*. MIT Press, 1998.

[118] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, pp. 130–137, 1980.

[119] D. Rafiei and A. O. Mendelzon, "What is this page known for? Computing web page reputations," in *Proceedings of the Ninth International World Wide Web Conference*, Amsterdam, The Netherlands, 2000.

[120] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis, "Automatic fragment detection in dynamic web pages and its impact on caching," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 5, pp. 859–874, 2005.

[121] A. R. Rao and P. Bhimasankaram, *Linear Algebra*. New Delhi: Tata-McGraw Hill, 1992.

[122] M. E. Renda and U. Straccia, "Web metasearch: rank vs. score based rank aggregation methods," in *Proceedings of the 2003 ACM Symposium on Applied Computing*, Melbourne, FL, USA, pp. 841–846, 2003.

[123] M. Richardson and P. Domingos, "The intelligent surfer: Probabilistic combination of link and content information in pagerank,," in *Advances in Neural Information Processing Systems 14*, Vancouver, Canada, pp. 1441–1448, 2002.

[124] E. Riloff, "Little words can make a big difference for text classification," in *Proceedings of the Eighteenth ACM International Conference on Research and Development in Information Retrieval*, Seattle, WA, USA, E. A. Fox, P. Ingwersen, and R. Fidel, Eds., pp. 130–136, 1995.

[125] M. Ruiz-Casado, E. Alfonseca, and P. Castells, "Automatic extraction of semantic relationships for wordnet by means of pattern learning from wikipedia," in *Proceedings of the Tenth International Conference on Applications of Natural Language to Information Systems*, Alicante, Spain, pp. 67–79, 2005.

[126] G. Salton and M. J. McGill, *Introduction to modern information retrieval*. New York: McGraw-Hill, 1983.

[127] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[128] G. Salton, "Developments in automatic text retrieval," *Science*, vol. 253, no. 5023, pp. 974–980, 1991.

[129] E. Sanchez, "Eigen fuzzy sets and fuzzy relations," *Journal of Mathematical Analysis and Applications*, vol. 81, no. 2, pp. 399–421, 1981.

[130] A. F. Smeaton, "Independence of contributing retrieval strategies in data fusion for effective information retrieval," in *Proceedings of the Twentieth BCS-IRSG Colloquium*, Autrans, France, 1998. [Online]. Available: http://www.bcs.org/server.php?show=ConWebDoc.4416

[131] M. Spitters, "Comparing feature sets for learning text categorization," in *Proceedings of International Conference on Computer-Assisted Information Retrieval (RIAO)*, Paris, France, pp. 1124–1135, 2000.

[132] B. S. Suryavanshi, N. Shiri, and S. P. Mudur, "Improving the effectiveness of model based recommender systems for highly sparse and noisy web usage data," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, Compiégne, France, pp. 19–22, 2005.

[133] J. L. Szwarcfiter and P. E. Lauer, "A search strategy for the elementary cycles of a directed graph," *BIT*, vol. 16, pp. 192–204, 1976.

[134] P. Tsaparas, "Link analysis ranking," Ph.D. dissertation, University of Toronto, 2004.

[135] V. N. Vapnik, *The nature of statistical learning theory*. New York: Springer-Verlag, 1995.

[136] "World wide web consortium," http://www.w3.org.

[137] A. Wald, *Sequential Analysis*. New York: John Wiley and Sons, 1947.

[138] "Webkb data set," http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/.

[139] S. M. Weiss, N. Indurkhya, T. Zhang, and F. Damerau, *Text Mining: Predictive Methods for Analyzing Unstructured Information*. New York: Springer, 2005.

[140] J. T. Welch, Jr., "A mechanical analysis of the cyclic structure of undirected linear graphs," *Journal of the ACM*, vol. 13, no. 2, pp. 205–210, 1966.

[141] "Wikipedia, the free encyclopedia," http://en.wikipedia.org/wiki/Main_Page.

[142] B. Wu and B. D. Davison, "Identifying link farm spam pages," in *Proceedings of the Fourteenth International World Wide Web Conference: Special Interest Tracks & Posters*, Chiba, Japan, pp. 820–829, 2005.

[143] S. Wu, F. Crestani, and Y. Bi, "Evaluating score normalisation methods in data fusion," in *Proceedings of the Third Asia Information Retrieval Symposium*, Singapore, 2006. [Online]. Available: http://www.cs.strath.ac.uk/~fabioc/papers/06-airs.pdf

[144] J. Xu and W. B. Croft, "Corpus-based stemming using coocurrence of word variants," *ACM Transactions on Information Systems*, vol. 16, no. 1, pp. 61–81, 1998.

[145] "Yahoo! Inc." http://www.yahoo.com/.

[146] "Yahoo! search index size," http://www.ysearchblog.com/archives/000172.html.

[147] F. Yamout, R. Demachkieh, G. Hamdan, and R. Sabra, "Further enhancement to Porter algorithm," in *Proceedings of the KI2004 Workshop on Machine Learning and Interaction for Text-based Information Retrieval*, Germany, pp. 7–24, 2004.

[148] Y. Yang, "A re-examination of text categorization methods," in *Proceedings of Twenty Second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, CA, USA, pp. 42–49, 1999.

[149] L. Yi and B. Liu, "Web page cleaning for web mining through feature weighting," in *Proceedings of Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, pp. 43–50, 2003.

[150] L. Yi, B. Liu, and X. Li, "Eliminating noisy information in web pages for data mining," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, D.C., USA, pp. 296 – 305, 2003.

[151] H. P. Young, "An axiomatization of Borda's rule," *Journal of Economic Theory*, vol. 9, no. 1, pp. 1–91, 1974.

[152] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[153] N. Zhong, "Impending Brain Informatics (BI) Research from Web Intelligence (WI) Perspective," *International Journal of Information Technology and Decision Making*, vol. 5, no. 4, pp. 713–727, 2006.

[154] N. Zhong, J. Liu, and Y. Y. Yao, "In search of the wisdom web," *IEEE Computer*, vol. 35, no. 11, pp. 27–31, 2002.

[155] N. Zhong, J. Liu, and Y. Y. Yao, "Envisioning intelligent information technologies through the prism of web intelligence," *Communications of the ACM*, vol. 50, no. 3, pp. 89–94, 2007.

List of Publications of the Author

[1] **B. L. Narayan**, C. A. Murthy, and S. K. Pal, "Topic continuity for web document categorization and ranking," in *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, pp. 310–315, 2003.

[2] **B. L. Narayan** and S. K. Pal, "Detecting sequences and cycles of web pages," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, Compiégne, France, pp. 80–86, 2005.

[3] **B. L. Narayan** and S. K. Pal, "A fuzzy web surfer model," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, Compiégne, France, pp. 120–123, 2005.

[4] S. K. Pal, **B. L. Narayan**, and S. Dutta, "A web surfer model incorporating topic continuity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 5, pp. 726–729, 2005.

[5] **B. L. Narayan** and S. K. Pal, "Distribution based stemmer refinement," in *Proceedings of the First International Conference on Pattern Recognition and Machine Intelligence*, Kolkata, India, pp. 672–677, 2005.

[6] **B. L. Narayan**, C. A. Murthy, and S. K. Pal, "Maxdiff kd-trees for data condensation," *Pattern Recognition Letters*, vol. 27, no. 3, pp. 187–200, 2006.

[7] **N. L. Bhamidipati** and S. K. Pal, "Comparing rank-inducing scoring systems," in *Proceedings of the Eighteenth International Conference on Pattern Recognition (ICPR06)*, Hong Kong, pp. 300–303, 2006.

[8] **N. L. Bhamidipati** and S. K. Pal, "Stemming via distribution-based segregation for classification and retrieval," *IEEE Transactions on Systems, Man and Cybernetics B*, vol. 37, no. 2, pp. 350–360, 2007.

[9] **N. L. Bhamidipati** and S. K. Pal, *Web Intelligence Meets Brain Informatics*. Berlin: Springer Verlag, 2007, ch. Fuzzy Web Surfer Models: Theory and Experiments, pp. 324–340.

[10] **N. L. Bhamidipati** and S. K. Pal, "Comparing scores intended for ranking," *IEEE Transactions on Knowledge and Data Engineering*, Revised and Submitted.