Strategic Polysemantic Search for Web Mining

A thesis submitted to the University of Hyderabad

in partial fulfillment of the requirements for the award of

Doctor of Philosophy

in

Computer Science

by

K. HIMA BINDU

School of Computer and Information Sciences

University of Hyderabad

Hyderabad – 500 046

India

June 2013

CERTIFICATE

This is to certify that the thesis entitled **"Strategic Polysemantic Search for Web Mining"** submitted by **K.Hima Bindu** bearing Reg. No. **08MCPC06** in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by her under my supervision and guidance.

The thesis has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Prof. C.Raghavendra Rao
Professor
School of CIS
University of Hyderabad
Hyderabad – 500 046

Prof. Arun K.Pujari
Dean
School of CIS
University of Hyderabad
Hyderabad – 500 046

DECLARATION

I, **K.Hima Bindu**, hereby declare that this thesis entitled **"Strategic Polysemantic Search for Web Mining"** submitted by me under the guidance and supervision of **Prof. C.Raghavendra Rao** is a bonafide research work. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma.

Date:

Signature of the student
Name: K. Hima Bindu
Reg. No: 08MCPC06

ACKNOWLEDGEMENTS

First of all I express my profound gratitude and deep regards to my supervisor, Prof. C. Raghavendra Rao. I'm grateful to him for all the motivating discussions on several Web and Data Mining problems. I admire his intuitiveness and his suggestions in directing my research towards the development of tools to address Query Ambiguity problem, which are discussed in this Ph.D. My gratitude towards him is very small when compared to the support and guidance received from him.

I gratefully acknowledge my DRC members Dr. Siba Kumar Udgata and Dr. Alok Singh for their valuable suggestions. I learned many useful insights from their interactions in my DRCs.

I would like to express my sincere gratitude to the Dean, School of Computer and Information Sciences, for his cooperation.

I am thankful to faculty of our school, whose interactions made my research complete. I also thank the technical and office staff of AI lab and our school for their cooperation.

My special thanks to my friends and research scholars in our school, especially Mr. P.S.V.S.Sai Prasad, Dr. K. Swarupa Rani, Dr.N. Naveen and Mr. B.Vikranth. Their encouragement and support is invaluable, I will forever be thankful to them.

I am extremely grateful to Mrs. C. Vijaya Lakshmi w/o Prof. C.Raghavendra Rao for her parental guidance and care. I am grateful for making the time I spent with you enjoyable and memorable.

I express my gratitude towards the Secretary and Principal of Ramachandra college of Engineering, for their motivation and encouragement. I acknowledge my colleagues for their cooperation.

The past five years have not been an easy ride, both academically and personally. I truly thank my family members for their constant encouragement.

It would not have been possible to write this doctoral thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here. Finally, I thank all of them.

ABSTRACT

Query ambiguity is an important and an interesting problem that search engines face. On an average, search engine queries are 2.4 words long, which, in most cases, does not reveal the intention of the searcher. Clustering of the search results is one of the ways to address this problem, classically known as Search Results Clustering (SRC).

The present study is an outcome of the aim to build a Meta Search Engine (MSE) for enhancing the web performance, addressing the issues associated with Query ambiguity. It is observed that the intermediate operations involved in Web browsing need to be soft real time, if not, hard real time. The components in this paradigm are centric around issues like Query ambiguity (Word sense disambiguation, Diversification of the results, Query Reformation, Search Results Clustering), Grouping (need to be addressed by clustering) for informative organization, Classification (Query classification, Content classification), Information encapsulation (dimension enhancement through Inferred Attribute for achieving compact models) and Query refinement and its consistency.

Analysis of this problem and the literature survey led the present study to give raise the following novel tools : Inferred Attribute based dimension enhancement for effective machine learning, A Decision Tree based on the Coefficient of Variation, A Greedy clustering approach, Consistency Analysis of Query Reformation. These have been integrated in the proposed architecture of MSE. These tools are also useful as stand alone Machine Learning or Data Mining tools.

Web search is in general an interactive activity, the proposed MSE assists the user in reformulating the query through the acquired knowledge (polysemantic) based on the search results till that point (through the tools developed). The integration aspects of this MSE prototype are provided and demonstrated.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

The Web is a huge collection of information with unorganized, unstructured and decentralized collection of web pages serving as data. Search engines play a key role in extracting relevant information from the Web. People use search engines to gather information that assist them for decision making in various disciplines and activities. Search engines receive a query from the user and use the keywords in the query to retrieve related web pages in the form of a list of search results. These search results are ordered based on the relevancy with the query posed by the user. The results retrieved and their order varies with the search engine used. The keyword based search feature available from search engines have limitations in retrieving relevant information. The limitations in indexing the dynamic and unorganized content on the Web and the linguistic features (synonymy and polysemy) of the language used in posing the query reduce the relevancy of the information in the search results.

The keyword based search and the ranking of the search results works well when the user can pose a well formed query matching the index of the search engine. However, users of Web often pose very short queries and it is difficult for the search engines to identify users interest and context in retrieving relevant information. The problem in forming appropriate queries is difficult in case of novice users and when the users are unfamiliar with topic they are querying about. In such cases, search engines usually return millions of results which may not be informative and relevant. Users have to sift through this long list of results and learn appropriate query terms from these results and try various reformed queries to satisfy their information need.

Techniques that can consolidate the search results information and enable the user to realize that the query can be reformed to gather focused information can provide better search experience for users. Hence, the aim of this thesis is to develop such techniques. The rest of the chapter gives overview of search engines, query ambiguity, polysemantic queries, problem of study, contributions made in this thesis and the organization of the thesis.

1.1 Web search

In today's world of information overload, people use Search Engines for almost every thing. Search Engines are ubiquitous, they act as filters for the wealth of information available on the Internet. Search engines usage is the second most popular web service, after e-mail [4] with 85% of Internet users utilizing search engines for their informational needs. The Web poses a great challenge for search engines due to enormous size of the Web, and due to the freedom to publish anything causing the web pages to be noisy, low quality and unreliable. Even though search engines are good at retrieving relevant pages, they usually return millions of search results. Sifting through the long list of search results is for relevant results is tedious, users normally search the first few tens of results and either give up or try other queries. The ambiguity present in query makes this problem severe, which is discussed in the next section.

1.2 Query Ambiguity

Finding relevant information from the huge collection of web pages is a herculean task. The Indexed Web contains at least 14.35 billion pages, as on Tuesday, 16 April, 2013 [118]. Even though the search engines are good at retrieving relevant results from the enormous collection of Web, they still suffer from Query Ambiguity problem.

Ambiguous query is a query that has more than one meaning [121] (e.g."apple", "tiger","jaguar"). Ambiguity stems from the context of the query, linguistic property of words such as synonymy and polysemy(i.e. a word having multiple meanings) and from the low average number of query words [71]. It is estimated that 16% of most frequent queries are ambiguous [115], hence query ambiguity poses challenge to the search engine developers.

1.2.1 Polysemantic Queries

Polysemy refers to the characteristic of natural language words having multiple meanings. Polysemantic queries need not be single word queries. 7% of ambiguous words in WordNet are multi-term words, while 39% of ambiguous entries in Wikipedia are multi-term words [115]. According to [69], there is a high probability to encounter queries that are ambiguous due to the many possible senses that a word can have.

Sample Polysemantic Queries

Few polysemantic queries with their possible meanings are listed below

apple can mean computer, phone, fruit

interpol can mean police, music

java can mean programming language, software, coffee, island

camel can mean animal, valley, software

snow leopard can mean animal, OS X

circuit can mean electronics, legal

1.2.2 Short Queries

Humans think in terms of concepts but the search is performed using words. Many times the query terms are ambiguous words, unable to fully represent the concept that the user has in mind. The intended meaning of such words is described by other words commonly occurring in the vicinity or context of these words [69]. However, on an average, search engine queries are 3.08 words long [124], which, in most cases, does not reveal the intention of the user.

1.2.3 Context

A query will have different meanings due to the context of usage. For instance, the word 'windows' has a meaning associated with Operating Systems in Computer Science environment, whereas in a building context, it means a opening in a wall and an aperture in the context of Optics. Search terms with different meanings in different contexts induce ambiguity while providing search results.

1.2.4 Synonyms

Words with same or similar meaning are called synonyms. For example, a synonym of *begin* is *commence*. Synonyms contribute to query ambiguity through vocabulary mismatch, the keywords posed in the query can be synonyms of the keywords used in the index of the search engine. When the keywords used in the query are not the same as those used by the search engine index, the retrieved results vary from the expected results.

1.3 Resolving Query Ambiguity

By resolving query ambiguity, the search engines can provide better search experience for users. Web directories like Open Directory Project(ODP) [96] or Yahoo Directory [39] took the lead in addressing query ambiguity. Web directories organize web sites by categories and subcategories. Search engines can use these categories to organize the search results. But, these Web directories are static and cover only a small part of the Web. It is reported that directory based systems are among the most ineffective solution [16] for resolving query ambiguity. Google used ODP till 2011 [123].

Semantic Information Retrieval(SIR) is another approach to solve the query ambiguity. It is performed by indexing and searching of concepts by means of Word Sense Disambiguation (WSD) rather than terms [92]. It relies on WordNet [140], hence suffers from its static structure and dearth of proper nouns. Many methods for SIR exist [93], however their applicability is debated [114].

A more popular approach to query ambiguity is Search Results Clustering (SRC) [21]. This is a post processing approach in contrast to the earlier methods. Given a query, it starts from the flat ranked list of results and clusters them based on the similarity among the results. This approach overcomes the limitations of the static and outdated information, as it works on the results returned from a search engine. This clustering is based on textual similarity of the results and senses, aspects or topics of the query. Some of the search results clustering methods work with search results alone [23, 24, 26, 33, 44, 73, 75, 80, 87, 95, 97, 105, 145–147], while others use external resources (search engine logs are used by [135], WordNet is used in [57], ODP used in [67] morphological lexicon for English [72] is used by [11], Google Web1T corpus [13] is used in [37, 94], WordNet and Wikipedia are used by [56], Wikipedia and TAGME [45] are used in [116]).

Diversifying the search results so that the top results displayed are different from each other support the user for narrowing the scope of user intention. This approach, called diversification of search results [2, 93], is applied by search engines like Google and Yahoo! up to some extent [93]. While this approach is useful to quickly find at least one result for relevant subtopics, retrieval of more information on specific subtopics of user interest is not facilitated [141].

Web Query Classification and Web Page Classification [10, 50, 62] aim at classifying the queries or web pages into predefined categories. These techniques help to improve quality of search results and resolve query ambiguity.

1.4 Motivation

A personalized search engine called SnakeT [44] has shown that a mutual reinforcement relationship exists between ranking and SRC. Many individual solutions are proposed for resolving query ambiguity problem. However, a unified machine learning approach which can offer subtopic retrieval and query reformation for query ambiguity resolution is not available. Customizing the search results with user interaction enhances the user's search experience by resolving query ambiguity. Awareness of hidden user intentions behind the query will facilitate to organize/personalize the search results. Extracting the hidden intentions should be made possible through machine learning approaches to overcome the subjectivity as well as to enhance the quality of the service. This motivated the present work to develop a context based personalized search engine to improve users search experience. Hence, the aim of this thesis is to come up with a meta search engine and to build/develop tools for the same.

The literature survey on web query classification and clustering search results [21, 50] motivated the present work to develop classification and clustering algorithms. The following tools are developed as part of this research: Inferred Attribute based dimension enhancement for effective machine learning, a Decision Tree based on the Coefficient of Variation, a greedy incremental clustering approach and consistency analysis of query reformation. These tools (except consistency analysis of query reformation) are also useful as stand-alone Machine Learning or Data Mining tools. These have been integrated in the proposed architecture of the Meta Search Engine.

1.5 Problem definition

Uncertainties in user queries due to the shortness, hidden intentions etc made the present study to consider the problem of

"Building a meta search engine with the support of apt machine learning tools and appropriate query enhancement".

The auxiliary problems addressed in this thesis are

- Search result clustering
- Classification of search results
- Query enhancement

And the critical issues addressed in this thesis are

- threshold fixing
- feature selection
- quantification of query enhancement

The following provides a brief account of tools developed and the functional aspects of the proposed meta search engine (MSE).

In response to the query submitted by the user, the proposed MSE extracts search results by using the APIs of search engines. These search results are clustered by using snippet information by using a novel clustering technique developed. These clusters are labeled based on the content of snippets of the clusters. These labels are used for enhancing the query to narrow down the search. Query enhancement evaluation method is developed to support the user in choosing the reformed query. Whenever user extracts additional web pages for the current query, the extracted search results are classified with the help of classifiers developed (Hybrid Classifier and Coefficient of Variation based Decision Tree), organized into folders. An interactive visualization module assists the user in interacting with the MSE.

Hence the auxiliary problems addressed in this thesis are search results clustering, classification of search results, query enhancement and validation method for query enhancement. And the critical issues associated with threshold fixing, feature selection, quantification of query enhancement are tackled.

The developed tools (clustering approach based on greedy incremental approach, Hybrid classifier and Coefficient of Variation based Decision Tree) are demonstrated and validated extensively by considering benchmark datasets to establish them as stand-alone data mining/machine learning tools and also suitable for the MSE.

The system design of the MSE is illustrated in Figure 1.1.

1.6 Major contributions

The taxonomy of the work is shown in Figure 1.2 along with the contributions. The contributions towards the thesis are highlighted with background color green and blue (green for the contributions suitable for SRC as well as stand alone Data Mining tools, blue for contribution suitable for SRC alone) briefly explained in the following:

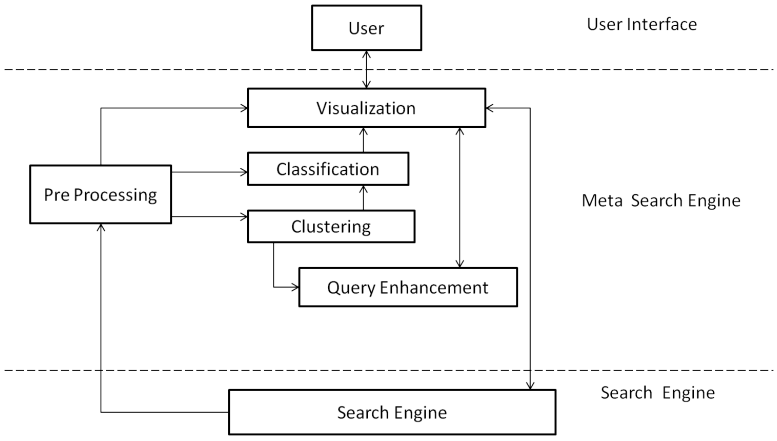

Figure 1.1: MSE system design

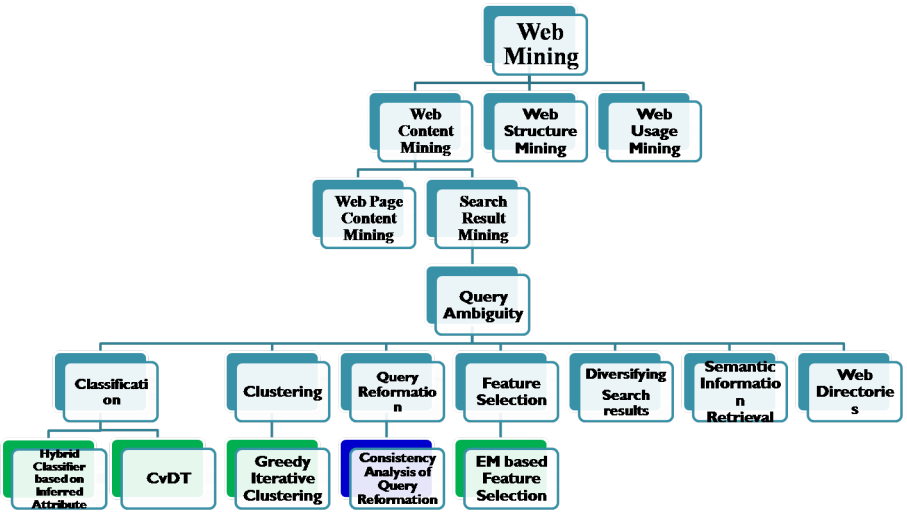

Figure 1.2: Taxonomy of the work

A meaningful dimension enhancement with inferred attributes is proposed to build faster and simpler classifiers. The inferred attributes induce a better dimensionality reduction. Reduct, which is a minimal set of attributes from Rough sets is used to achieve dimension reduction.

A less computationally complex attribute selection measure "CvGain" based on Coefficient of Variation is proposed. The decision tree can be built in significantly less time using the proposed method when compared with other decision trees. This decision tree can handle continuous decision attributes and is suitable for induction of the decision tree from distributed data.

A novel clustering algorithm is proposed to handle both numeric as well as categorical data in two modes: object centric and feature centric. A feature selection method has been evolved by using Expectation Maximization algorithm, as a pre-processor for feature centric clustering algorithm used for the clustering

of search results.

Two statistical measures, Rank correlation coefficient and coefficient of contingency have been tailored for evaluating modified query in contrast to the initially posed query. These measures are instrumental for suggestive query enhancement.

The proposed Meta search engine is a hybrid and appropriate integration of the above contributions, for providing localized search results to address the polysemantic ambiguity.

These contributions led to the following publications.

1.6.1 List of Publications

1. K.Hima Bindu, P.S.V.S.Sai Prasad, and C.Raghavendra Rao. "Hybrid Decision Tree based on Inferred Attribute". In *Proceedings of the First Amrita ACM-W Celebration on Women in Computing in India*, A2CWiC'10, ACM, New York, NY, USA, pp 61-67. [Indexed by DBLP]

2. P.S.V.S.Sai Prasad, K.Hima Bindu, and C.Raghavendra Rao. "Incremental Learning in AttributeNets with Dynamic Reduct and IQuickReduct". In *Proceedings of the 6th international conference on Rough sets and knowledge technology*, RSKT'11, Springer-Verlag Berlin, Heidelberg 2011. [Indexed by DBLP]

3. K.Hima Bindu, K. Swarupa Rani, and C.Raghavendra Rao. "Coefficient of Variation based Decision Tree (CvDT)". International Journal of Innovative Technology and Creative Engineering, ISSN:2045-8711, June 2011 Issue Vol.1 No.6. [Indexed by Google Scholar]

4. K.Hima Bindu, P.S.V.S Sai Prasad, and C.Raghavendra Rao. "Consistency Analysis of Query Reformation". In *Proceedings of Emerging Trends in Soft Computing*, NCETSC'11, Pune, India. Article 4, Excel India Publishers, ISBN 978-93-80697-56-7.

5. K.Hima Bindu, and C.Raghavendra Rao. "Association Rule Centric Clustering of Web Search Results". In *Proceedings of Multi-disciplinary Trends in Artificial Intelligence*, MIWAI'11, Lecture Notes in Computer Science, 2011, Volume 7080/2011, pp 159-168. [Indexed by DBLP]

6. K.Hima Bindu, and C.Raghavendra Rao. Search Result Clustering through Expectation Maximization based Pruning of Terms". In *Proceedings of Soft Computing for Problem Solving*, SOCPROS'12, December 28-30, Computational Intelligence and Complexity, Springer, ISBN 978-81-322-1601-8.

1.7 Organization of the Thesis

Chapter 1 is the introduction of the thesis. This chapter contains a review of Web search and query ambiguity problem. The motivation for developing MSE is put forward. The contributions of the thesis and organization of the thesis are presented.

Chapter 2 provides the basic definitions and the literature survey. The query ambiguity resolution approaches are presented with their merits and limitations. An overview of clustering search engines and personalized search engines is provided. The evaluation methods for search results clustering are described in detail. An overview of the data mining functionalities used in this thesis, namely classification and cluster analysis is presented. As the search results contain textual data, the fundamentals of text mining are described.

Chapter 3 discusses the proposed Inferred Attribute based dimension enhancement for effective machine learning. This is a hybrid approach based on Regression, Rough sets and a Classifier aiming at effective dimensionality reduction, henceforth a classifier with less time complexity. It focuses on dimension enhancement by embedding Inferred Attribute into the decision table, named as Augmented Decision Table (ADT). An effective minimal set of attributes (reduct) is derived by employing Rough set theory on ADT. The projected ADT based on the reduct is used for building a classifier. The philosophy is demonstrated by considering popular functional forms for Inferred Attribute like Linear Regression and Quadratic Regression and by adopting CART as the classifier. It is observed that inclusion of Inferred Attribute reduces the size of the reduct significantly without compromising the accuracy. This work is published in *proceedings of A2CWiC'10*.

Chapter 4 puts forth a Decision Tree based on the Coefficient of Variation. The decision tree is developed using a novel splitting criteria based on Coefficient of Variation, named as Coefficient of Variation Gain (CvGain). Empirical analysis presented on standard data sets reveal its merits in reducing computational cost and time. This work is published in International Journal of Innovative Technology and Creative Engineering. CvDT on the reduced augmented decision table has achieved significant performance of 94% on the WebKB text data set in comparison to the existing approaches.

Chapter 5 presents the proposed greedy incremental clustering approach. It can be feature centric or object centric. The features are extracted from title and snippet of the search results. The clustering algorithm greedily picks up the data objects/features to initiate a cluster, the objects in its neighborhood are added repeatedly in the growing phase of the cluster. This clustering approach is named

as *Snowball clustering* due to its nature. When the cluster cannot grow further, the next object/feature is picked up in greedy manner to initiate the next cluster. In feature centric approach, the objects are grouped based on the features clustered in each iteration of the growing phase of the cluster. A label generating module which can generate meaningful and human readable labels is developed. The developed clustering method and label generating module are demonstrated on the web search results based on Association Rules. This work is published in the *proceedings of MIWAI'11*. The findings of [47] has shown that frequency based feature ranking is not suitable for SRC. Hence Expectation Maximization based adaptive feature pruning method is proposed. The features not pruned by this method are considered as relevant features. Knowledge preserving capabilities of these relevant features and Snowball clustering method based on these relevant features is demonstrated. This preprocessing approach is published in the *proceedings of SOCPROS'12*.

Chapter 6 provides a schematic analysis of consistency aspects of query reformation techniques used in search engines. Study of various statistical measures elicit the similarities of web queries and effectiveness of the query reformation. This work is published in the *proceedings of NCETSC'11*.

Chapter 7 details the proposed Meta Search Engine (MSE). The prime concern in developing this MSE is to provide a better search experience in case of broad and ambiguous queries. When the user submits a query, the top N (MSE parameter) search results are gathered from conventional search engine(s), clustered into subcategories using the proposed greedy clustering approach. By observing the labels and the content of the clusters, the user can quickly focus on relevant subtopic of the query. The MSE allows the user to explore more results obtained from conventional search engine(s) and classify them according to the subtopics using the developed classifiers. The query enhancement module is provided to enable the user to reformulate the base query and refresh the results and the clusters.

The thesis is concluded in **Chapter 8** with future directions. The main challenge in human system communication, vocabulary mismatch, is addressed in the thesis. The proposed MSE is demonstrated to enhance the search experience of the user in case of broad and ambiguous queries. The proposed MSE applies customization based on user's choice.

CHAPTER 2

Literature Survey

This chapter gives an overview of the techniques useful to develop the tools needed by the meta search engine developed in this thesis. Query ambiguity problem and various solutions for it are discussed in Section 2.1, the search result clustering approach is discussed in detail in this section. Existing clustering engines and their benefits are provided in Section 2.2. Section 2.3 presents the evaluation metrics for validating the search result clustering algorithms. Personalized search engines are described in Section 2.4. An overview of relevant data mining tools, namely classification and clustering methods are described in Section 2.5.

2.1 Overview of Query Ambiguity Resolution

Search engines usually return a long list of search results in response to a query. These are sorted according to the relevance to the query posed. Users will not check the results beyond first few pages of search results (usually ten results appear in each page). The results retrieved and the rank order keep varying with the search engines, as these are guided by the frequency of the crawler and the ranking algorithm used by the search engines. Search engines compare the query terms against the inverted index built in parallel to crawling the Web. The web pages matching the query terms in the index are returned based on the ranking algorithm. Many times query terms are ambiguous and users pose short queries [124], hence, they match vast number of web pages with various concepts/topics. The linear list of search results may push the results which are useful to the user to such a position that user may not have enough patience and time to explore till that result. Users normally explore first few pages only. This problem is called query ambiguity, which stems from a property of natural languages called polysemy (a word having multiple meanings is polysemous) and due to insufficient number of query terms for identifying the context of the query.

The solutions for addressing query ambiguity are discussed in the following subsections.

2.1.1 Web directories

Web directories list web sites organized by category and subcategory (mostly by human editors). Yahoo! Directory [39] and the Open Directory Project (ODP) [96] are two well known web directories. When a query is posed, the results are listed according to categories and subcategories. Web directories attempt to resolve query ambiguity by identifying categories. However, as most of the web directories are built by human editors, they suffer from static nature requiring manual updates to cover new web pages and new categories [94].

2.1.2 Semantic Information Retrieval

Semantic Information Retrieval (SIR) performs search on word senses/concepts. SIR addresses synonymy and polysemy. Word Sense Disambiguation [92] is used by SIR. However, word sense disambiguation depends on dictionaries like WordNet. This results in a poor coverage of real Web queries [115]. In addition, dictionaries (e.g., WordNet) are static and have low coverage.

2.1.3 Search Result Clustering

Search Result Clustering (SRC) is a more popular solution for query ambiguity. It is a post retrieval technique and clusters the search results returned by conventional search engines to identify the thematic groups. SRC is useful when the user is interested in retrieving multiple documents relevant to each subtopic of polysemous queries. This approach does not suffer from the static nature of the earlier approaches as it works on the output of search engines. Many algorithms for SRC are proposed in the literature, see [22] for survey. According to the survey, SRC algorithms are categorized as data-centric, description-aware, and description-centric.

Data-centric approaches use conventional data clustering algorithms and produce some kind of textural representation. Scatter-Gather [33] is the pioneering algorithm and treated as a predecessor of all SRC algorithms. It performs an initial clustering of a collection of documents into a set of k clusters (a process called scattering), when user selects a cluster it reclusters the documents (the gathering step). It used agglomerative hierarchical clustering with an average-link merge criterion [42]. Other data-centric approaches are Lassi [82] using agglomerative hierarchical clustering on pair of words with lexical affinity, TRSC [95] based on Rough sets [101]. The only problem with these methods is arriving at textual

cluster representation.

Description-aware clustering methods try to find clusters with human readable labels. Suffix Tree Clustering (STC) [145] is the seminal work in this category. STC uses phrases to form base clusters which are subsequently merged based on cluster overlap. HSTC [87], Carrot [137], SnakteT [44] are follow up approaches with improvements on STC.

Description-centric uses the "description comes first" strategy of Vivismo [134], a commercial meta search engine which organizes the search results. Lingo [97, 98] uses Singular value decomposition (SVD) on frequent phrases obtained through Suffix Arrays [85]. SHOC [149] is similar to Lingo algorithm. SRC [147] uses salient phrases and regression, Discover forms concepts based on noun phrases and salient keywords [75], CREDO [23] is based on concept lattices. KeySRC [11] identifies 'keyphrases' from generalized suffix tree and clusters the 'keyphrases' by hierarchical agglomerative clustering. Word Sense Induction (WSI) is used for automatic discovery of word senses in [92]. Given a query, word senses of it are acquired by using a text corpus. Clustering is performed on the basis of word senses.

The SRC methods can also be classified based on the usage of external resources. They can work only with the search results or use external resources in addition to the search results. Recently, algorithms using external resources like ODP taxonomy, Wikipedia, WordNet etc are developed. These approaches generated clusters with good quality, but more complex when compared to methods that do not use external resources. In the following, the recent approaches are presented along with the external resource used.

ODP classifier to map search result URLs to ODP categories is developed in [67]. But, as all search results cannot be mapped to ODP categories, the unmapped results are categorized with a centroid-based classifier.

KeySRC is a Keyphrase-Based algorithm [11]. It uses Generalized Suffix Trees (GST). However, it has few unique features like using POS for GST pruning, employing a dynamic cut-off level of the clustering dendogram, and choosing labels that maximize the cluster coverage.

Word Sense Induction is used to Improve Web Search Result Clustering in [92]. They used Google Web1T corpus [13] to determine a set of co-occurring words, which were used to construct co-occurrence graphs for identification of word senses. Each result's bag of words are intersected with the senses, and the sense with largest intersection is the associated cluster.

Optimal Meta Search Results Clustering of [24] uses multiple search results

clustering algorithms' outputs to generate a single output. They presented the empirical justification for meta SRC and the meta SRC is proposed as an optimization problem. An objective function measuring the probabilistic concordance between the clustering combination and the single clustering was solved by using soft computing methods. This method did not use any external resource.

A method for identifying and ranking possible categories of any user query, based on the meanings and common usages of the terms and phrases within the query is proposed in [56]. They used WordNet and Wikipedia to recognize phrases and to determine the basic meanings and usages of each term or phrase in a query. Likelihood in capturing the query's intention guides the category ranking. Their work cannot be compared with the rest of the works as the data set used by them is different from the rest of the works.

Maximum Spanning Trees to identify the word senses (hence sense clusters) on the co-occurrence graph is built by [37], in similar way to [94]. The search results are clustered based on their semantic similarity to the induced word senses.

Topical Clustering of Search Results proposed by [116], used topic annotators (TAGME of [45]) to construct graph of topics. The edges of this graph are weighted based on Wikipedia liked structure. This graph happens to be an enhancement of term-document matrix and reduces its sparseness. A topical decomposition of this graph is performed based on disjoint subsets of the topics. They have considered the number of clusters to be 10 and it is the number of subsets (each subset is a cluster).

Challenges

SRC has new challenges when compared to traditional clustering algorithms, they are outlined in the following:

1. *Input data:* The SRC engines use title and snippet of the search results to perform the clustering. This information is meager and may not be informative for the clustering process.

2. *Response Time:* The Clustering Search Engines must have quick response times as this process is an on-line activity. The search results acquisition usually takes a long time, thus the clustering process must execute in parallel to the search results acquisition.

3. *Unsupervised learning:* Clustering is an unsupervised task. However, some clustering methods need the number of clusters to perform the clustering.

SRC has to work without such knowledge and has to form ephemeral clusters based on the search results obtained for a given query. Further, the clusters can be overlapping.

4. *Meaningful and human readable cluster labels:* Traditional document clustering algorithms use centroid to represent the clusters. However this is not suitable for SRC, as the cluster labels should indicate the contents of the cluster.

5. *Overlapping clusters:* A search result can appear in multiple clusters unlike the traditional clustering approaches.

Advantages

The following is the list of benefits offered by SRC.

1. *Subtopic retrieval of polysemous queries:* The subtopics of polysemous query become the cluster labels, hence user can quickly get the knowledge of the related topics of the query.

2. *Overcoming the information overload:* Typically search engines retrieve millions of search results, user is provided with the consolidated knowledge of first few hundreds of search results in a single page.

3. *Topic exploration:* All the search results of a topic are grouped and available at a single place. Thus the user can quickly explore all the web pages of topic.

4. *Suitable for dynamic and unknown domains:* When the user is unaware of vocabulary of an unknown domain, SRC facilitates learning of appropriate terms.

Applications

SRC is suitable for search using mobiles due to the challenges posed by small-screen, limited input and low bandwidth. SRC has the potential of minimizing the amount of information transmitted and displayed, as well reducing tedious user actions using mobiles such as scrolling and keyword entry.

In addition to web search, SRC can also be used for clustering the library search results, Medline, Twitter and blogs' snippets.

The concept (or entity) recognition that multiple search engines are working on (for finding e.g., people, products, books) also seems related to clustering of search results [143].

2.1.4 Query and Web Page Classification

Web Query Classification and Web Page Classification [10, 50, 62] aim at classifying the queries or web pages into predefined categories.

Web query classification assigns a Web search query to one or more predefined categories, based on its topics. Search results are classified into different predefined categories. This method suffers from challenges like *short and ambiguous queries* and *dynamic* nature (meaning changes with time and context) of queries in contrast to document classification. KDD cup 2005 [32] is the dataset released for KDD cup competition in 2005. Various researchers have attempted to address this problem using query enrichment techniques and search engine log.

Web page classification assigns web pages (web sites) to predecessor categories. For example, automated online web page classification is available from [99]. Techniques based on Web page content and structure are available in the literature.

These methods suffer from the static and human specified categories.

2.1.5 Diversification

Diversification of search results [2, 93], is applied by search engines like Google and Yahoo! up to some extent [93]. This method tries to rank the search results such that the top ranked results belong to different topics. While this approach is useful to quickly find at least one result for relevant subtopics, retrieval of more information on specific subtopics of user interest is not facilitated [141].

2.2 Clustering Search Engines

Clustering search engines automatically cluster results into categories/topics based on the phrases contained in search results. The categories bear human readable label and are normally presented in folder tree interface.

Clustering search engines offer a complementary view to the flat ranked list of results commonly returned by search engines. Through this users can quickly learn the topics related to the query and explore the results related to the topics.

Figure 2.1 serves as an example.

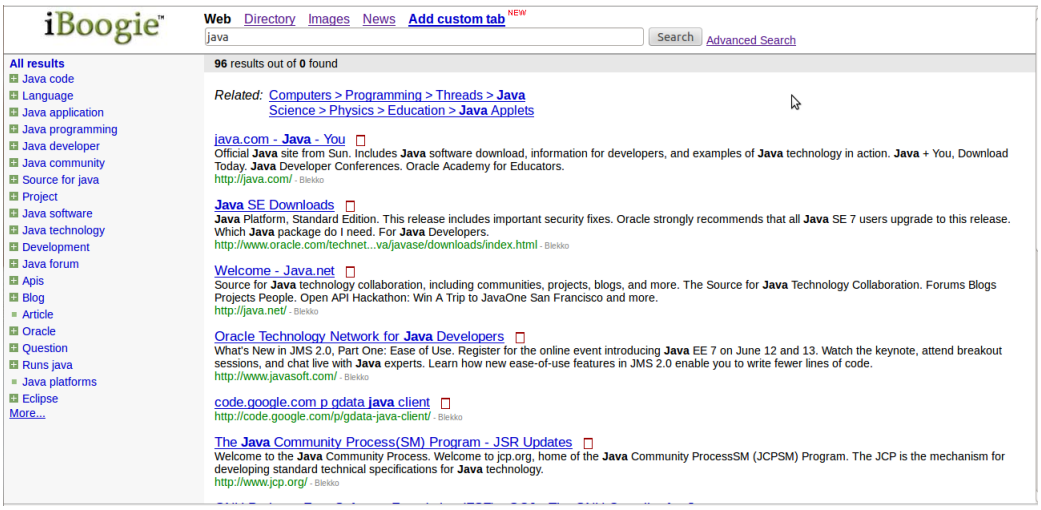

Figure 2.1: Web interface of iBoogie

The following is a list of some of the clustering search engines:

carrot search accessible at http://search.carrot2.org/stable/search

clusty accessible at http://yippy.com/

iBoogie accessible at http://iboogie.com/

query server accessible at http://queryserver.dataware.com/web.htm

dogpile accessible at http://www.dogpile.com/

polymeta accessible at http://www.polymeta.com/

helioid accessible at http://www.helioid.com/

webclust accessible at http://www.webclust.com/

The benefits of understanding the meanings of short text and clustering the concepts/topics of search results are listed as the following in [1] from the perspective of search engines:

- Help in guessing at concepts behind a piece of text. These concepts might be shown to a searcher during a search to help them better understand the meaning behind the text.
- Enabling the search engine to compare words and concepts found in a document and in a query. This can help the search engine come up with an information retrieval scoring function to help rank web pages in search results based upon those concepts.

- Expanding search results to include related words and concepts in a search by looking at clusters of potential results related to different concepts that might include a specific word. For instance, a search for the word *jaguar* could mean the *car*, the *animal*, or the *NFL football team*. Clusters created around each of these "concepts" associated with the term could lead the search engine to show a certain percentage of results covering the different concepts, and making sure of diversity in those results.

- Comparing the relationship between words and concepts on a web page and in an advertisement. This can stand in as a proxy for how well an advertisement might perform when displayed on a certain web page. For example, an advertisement for a jaguar car on a page about jaguar cats may not be very effective.

- Comparing the relationship between words and concepts in a query and in an advertisement. This can provide an idea of how well an advertisement might do on a search result page for a specific query.

- Comparing the relationship between words and concepts from different web pages. This can tell the search engine how far apart conceptually two pages might be when they are clustered together as similar documents.

- Classification of pages, and filtering of some kinds of pages, based upon which clusters words (that might be used within queries) tend to appear within.

- Generalizing a search query to retrieve more results, by looking at the clusters that the query terms appear within and parent concepts for those clusters.

- Identifying whether a word is a misspelling of another word by looking at the concepts related to each of those two words. For example, flicker is conceptually related to lights and flames, and Flickr is conceptually related to photographs. Theres a good probability that flickr is not a misspelling of flicker.

2.3 Evaluation metrics for Search Results Clustering

Clustering algorithm's performance is normally measured by internal measures (based on the information present in the data set itself) and external measures (comparison against ground truth). However, SRC validation requires verification of improvement in the retrieval performance. The following sections present the validation techniques for SRC.

2.3.1 Comparison against Ranked List

As SRC is proposed to overcome the limitations of plain search engines, it needs to evaluate whether SRC improves the retrieval performance over flat ranked lists. The classification oriented measures like Precision and Recall can be used after linearization of the clustered results. The results of a high density cluster or optimal cluster can be flattened for this purpose [21]. According to [77], a simple interactive method is more effective for linearization. The reach time and Subtopic Reach Time measures (discussed below) assume that a cluster label will allow the user to choose the right cluster. Hence, these measures provide an upper bound on the true retrieval performance.

Reach Time

An analytic method based on reach time is proposed in [98]: it models the time taken to locate a relevant document in the hierarchy. When s is the branching factor, d is the number of levels that must be inspected, $p_{i,c}$ is the position of the i^{th} relevant document in the leaf node of the clustering approach, $p_{i,r}$ is the position of the i^{th} relevant document in the ranked list; reach time of i^{th} document is

$$rt_{clustering} = s \times d + p_{i,c} \tag{2.1}$$

The corresponding reach time of ranked list is

$$rt_{rankedlist} = p_{i,r} \tag{2.2}$$

The averaged reach times of the set of relevant documents can be smaller than $rt_{rankedlist}$. The range of reach time is 1 to the number of search results (N) and smaller values are preferred.

Subtopic Reach Time

This measure is defined as the mean, averaged over the querys subtopics, of the smallest of the reach times associated with each subtopics relevant results. It is proposed by [20]. For n subtopics, the Subtopic Reach Time of ranked list is defined as below:

$$SRT_{rankedlist} = \frac{\sum_{i=1}^{n} \min_j(P_{i,j})}{n} \tag{2.3}$$

where $P_{i,j}$ is the position of the j^{th} relevant result of subtopic i. Hence average of the first relevant result position of each relevant subtopic is taken.

For clustered list, reach time takes into account both the cluster labels that

must be scanned and the snippets that must be read. The reach time of a clustered result is given by the position of the cluster in which the result is contained (c) plus the position of the result in the cluster (r):

$$SRT_{clusterlist} = \frac{\sum_{i=1}^{n} \min_j (c_{i,j} + r_{i,j})}{n} \tag{2.4}$$

The range of SRT is 1 to the number of search results (N), and smaller values reveal that SRC is effective. SRT can also be used to compare systems which work on the same set of search results. It treats all subtopics as equal, does not attach importance to popular subtopics.

Usage of Search Engine Logs

By comparing the search engine logs to clustering engine logs, [146] proposed that we can compare these approaches and avoid the requirement of a test collection with specified relevance judgments. Few metrics for this comparison are the number of documents followed, time spent and click distance. However, interpretation of user logs is difficult as it involves multiple users and different search tasks.

Usability tests/User studies

Conducting user studies is a viable alternative to automated evaluation of SRC methods. These are subjective tests while the earlier are objective measures. The user (i.e. subject) performs some kind of information seeking task with the systems being compared, the user session is recorded, and the retrieval performance is typically evaluated measuring the accuracy with which the task has been performed, and its completion time. Such studies are especially useful to evaluate inherently subjective features or to gain insights about the overall utility of the methods being tested [20]. Usually, subjects of intermediate web ability are made to participate in the experiment and they were assigned a task list. Several studies [20, 25, 44, 70, 128] have performed user studies to evaluate improvement in the search experience. The search performance and satisfaction level is studied in [106], with and without the aid of clusters and hierarchies. This study used a client logging software [120] to record each participants search process. Mechanical Turk (AMT - Amazon Mechanical Truck) [31] is used in [116] to generate the human ratings. The drawbacks of this methodology are that the tests are not repeatable/replicable, no standards for verification of the user study, dependency on subjects ability and bias. The user studies reported in the literature are favorable to clustering engines.

2.3.2 Quality of Cluster Labels

Each cluster label indicates the contents of the cluster; hence, meaningful labels are required for user exploration. Cluster labels can be assessed by verifying the keywords in the cluster labels with the manually assigned topic labels. Relevance of labels can be assessed manually ([122, 147]) or automatically ([76]) by measuring the labels relationship with clusters content. Salient phrase ranking is proposed by [149], it measures precision of list of labels associated with the clusters, assuming that relevance of labels has been manually assigned for each topic.

Precision at top N is used in [44]

$$P@N = \frac{M@N}{N} \tag{2.5}$$

where $M@N$ is the number of labels which have been manually tagged relevant among the N top-level labels. This measure reflects the user behavior for cluster hierarchy navigation. N values beyond ten were not considered, as users do not like to browse a wider cluster hierarchy.

2.3.3 Subtopic Retrieval

For a given query, we need to assess the SRC algorithms ability to retrieve the subtopics. Two measures are proposed for this purpose; first, $kSSL$, checks whether all documents relevant to the subtopics are retrieved; second, S-recall@K, checks the number of subtopics retrieved.

$kSSL$

To evaluate the retrieval performance of SRC, Subtopic Search Length under k document sufficiency ($kSSL$) is proposed in [11]. It measures the average number of items (labels or results) that must be examined before finding a sufficient number (k) of documents relevant to any of the query's n subtopics. If k documents could not be found with the clustering approach, then the user switches back to the ranked list. Hence, this measure models the users search behavior. For a ranked list, the value of $kSSL$ is simply given by the mean of the ranks of the k^{th} results relevant to each subtopic in the ranked list associated with the query:

$$kSSL_{list} = \frac{\sum_{i=1}^{k} p_{i,k}}{n} \tag{2.6}$$

Where $p_{i,k}$, is the rank of the k^{th} result relevant to subtopic i.

For clustered results, $kSSL$ definition involves both cluster labels that must be scanned and the snippets that must be read. Clusters whose labels are relevant to the subtopic at hand are considered. This separates $kSSL$ from the other methods which usually assume that the user is able to select the relevant documents, irrespective of the cluster labels. With these considerations, the formula is a sum of three terms: the rank of the last cluster of the m clusters with a relevant label that were visited before retrieving k results (denoted $c_{i,k}$), plus the sum of the cardinalities of the first $(m-1)$ visited clusters, plus the rank of the k^{th} relevant result in cluster $c_{i,k}$, (denoted $r_{i,k}$). Formally:

$$kSSL_{clusters} = \frac{\sum_{i=1}^{n}(c_{i,k} + \sum_{j=1}^{m-1} |c_{i,j}| + r_{i,k})}{n} \tag{2.7}$$

When the clusters with relevant labels exhaust before finding k relevant documents, the full ranked list of documents have to be considered. So, the number of search results that need to be considered has to be added to the above summation.

The minimum value of $kSSL$ depends on the number of subtopics. The topics with more subtopics will have a higher minimum value for $kSSL$, and its value increases with k. A computationally intensive procedure to normalize search lengths over the number of subtopics is proposed in [148]. By considering the weighted average of subtopics, popular subtopics can be given more importance.

S-recall@K

A measure of diversification, subtopic recall-at-K (S-recall@K proposed by [148], it can be used to evaluate the sub topics retrieved. S-recall@K is given by the number of different subtopics retrieved for query q in top K results returned:

$$S - recall@K = \frac{|\bigcup_{i=1}^{K} subtopics(r_i)|}{M} \tag{2.8}$$

where $subtopics(r_i)$ is the set of subtopics manually assigned to the search result and M is the number of subtopics for query q. This measure is suitable for systems returning ranked lists, so the clusters have to be flattened to a list as given by [94] and [37].

2.4 Personalized Search Engines

Personalized search combines contextualization and individualization to normal web search [103]. The search engines like Google and Yahoo! build the user

profiles based on the user preferrences and activities (history of user interaction). The user profiles guide the personalization.

Personalized search engines are criticized for the following reasons:

- Decreased likelihood of finding new information. The results are biased towards what the user has found already.
- Introduces privacy concerns.

2.5 Overview of relevant Data Mining Tools

The data mining tools used for building the MSE are discussed in this Section. A supervised learning method called Classification and an unsupervised learning method called Clustering are presented in the following sections along with their performance evaluation methods.

2.5.1 Classification Tools

Classification is the task of assigning class label to a data object. It is a supervised learning task, it starts with building a model using the dataset for which class assignments are known. The model is used to to accurately predict the class label of a new data object. There are many techniques for building classification models:

- Decision tree
- Rule-based classifier
- Nearest-neighbor classifier
- Bayesian classifier
- Artificial neural network
- Support vector machines

Decision tree classifier is simple and widely used classification technique. As two classifiers were developed in this thesis using decision trees, a literature survey on decision trees is provided in the following subsection.

Decision trees

A decision tree is a flowchart-like tree structure, where each internal node (nonleaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label [53].

Given a tuple, X, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules.

Decision trees are popular for the following reasons [53]:

- Simple, fast and human interpretable
- Does not require parameter setting
- Does not require any domain knowledge
- Can handle high dimensional data
- Have good accuracy in general
- They are the basis of several commercial rule induction systems

The popular decision tree algorithms are ID3 [107], C4.5 [110] and CART [14]. These follow a greedy approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner. Most algorithms for decision tree induction also follow such a top-down approach, which starts with a training set of tuples and their associated class labels. The training set is recursively partitioned into smaller subsets as the tree is being built. In general, may decision trees can be constructed from a given set of attributes. While some of the trees are more accurate than others, finding the optimal tree is computationally infeasible because of the exponential size of the search space [126].

ID3 [107], C4.5 [110] and CART [14] follow Hunt's algorithm [40] for building the decision trees. Hunt's algorithm grows a decision tree in a recursive fashion by partitioning the training records into successively purer subsets. Let D_t be the set of training records that are associated with node t and $y = \{y_1, y_2, ..., y_c\}$ be the class labels. Then Hunt's algorithm is the following.

1. If all records in D_t belong to the same class y_t, then t is a leaf labeled as y_t.
2. If D_t contains records that belong to more than one class, use an attribute test condition to split the data into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed

to the children based on the outcomes. The algorithm is then recursively applied to each child node.

Often, decision tree algorithms also include a pruning phase to alleviate the problem of overfitting the training data. Either pre-pruning or post-pruning techniques are used for this purpose [53].

Classifier Performance Measures

To measure a classifier performance, the data is divided into training data set and test data set. The classifier is built on training data set and its performance is assessed on test data. Usually 10-fold cross validation is employed to remove the bias in the training and test data sets. A confusion matrix [53] is built by comparing the predicted class against the ground truth.

Accuracy is a commonly used measure to find a classifier performance. The **accuracy** of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. The **error rate** or **misclassification rate** of a classifier, M, is $1 - Acc(M)$, where $Acc(M)$ is the accuracy of M. Using the confusion matrix, measures like precision, sensitivity and specificity are computed [53].

2.5.2 Clustering Tools

Clustering is a process of partitioning a set of objects (or data elements) into subsets, called clusters, such that an object belonging to a cluster is more similar to objects belonging to the same cluster than to objects belonging to other clusters. Partitioning is based on a (dis)similarity measure that is always a pair-wise measure. Clustering is a form of unsupervised learning compared to classification (or categorization) that is based on predefined classes (or categories).

A survey of the core concepts and techniques in the cluster analysis is available in [61].

Types of Clustering Algorithms

Numerous clustering algorithms exist in the literature, the most prominent algorithms can be broadly classified as the following [53]:

1. Partitioning methods

2. Hierarchical methods
3. Density based methods
4. Grid based methods
5. Model based methods

All the clustering algorithms try to minimize intra cluster distances and maximize inter cluster distances. Depending on the data, various distance measures are used to compute these similarity and dissimilarity among the data objects.

Evaluation of Clustering Algorithms - External Measures

These are used to measure the extent to which a clustering algorithm matches a pre specified external structure (ground truth or gold standard). These measures are supervised in nature [126].

Classification Oriented measures

These are used to measure the degree to which predicted cluster labels correspond to actual class labels [126]. The following subsections use the notations: m_i is the number of objects of cluster i and m_{ij} is the number of objects of class j in cluster i. L is the number of classes. K is the number of clusters and m is the total number of data points. The probability that cluster i belongs to class j is $p_{ij} = m_{ij}/m_i$.

1. *Entropy:* Entropy measures the degree to which each cluster consists of objects of a single class. For each cluster i, its class distribution is calculated as p_{ij}. The entropy of each cluster is computed as

$$e_i = -\sum_{j=1}^{L} p_{ij} \log_2 p_{ij} \tag{2.9}$$

   The total entropy for the set of clusters, e, is the weighted sum of each cluster entropy:

$$e = \sum_{i=1}^{k} \frac{m_i}{m} e_i \tag{2.10}$$

   The range of e is 0 to 1.The clustering algorithm is preferred when it has the minimum entropy, near to 0.

2. *Purity:* Purity is a measure of the extent to which a cluster contains objects of a single class. Using the same notations of previous subsection, the purity of a cluster i is $p_i = \max_j p_{ij}$, the overall purity of the

clustering is

$$Purity = \sum_{i=1}^{k} \frac{m_i}{m} p_i \tag{2.11}$$

The values of $Purity$ can vary from 0 to 1. Bad clusterings have purity values close to 0, a perfect clustering has a purity of 1.

3. *Precision:* It is the fraction of a cluster that consists of objects of a specified class. The precision of cluster i with respect to class j is $P_i = p_{ij}$. The total precision of the clustering is

$$P = \frac{\sum_{i=1}^{k} P_i m_i}{\sum_{i=1}^{k} m_i} \tag{2.12}$$

The range of P is 0 to 1. Higher values are preferred for better clustering, the best possible value is 1.

4. *Recall:* It is the extent to which a cluster contains all objects of a specified class. The recall of cluster i with respect to class j is, $R_j = m_{ij}/m_j$. The total recall of the clustering is

$$R = \frac{\sum_{j=1}^{L} R_j m_j}{\sum_{j=1}^{L} m_j} \tag{2.13}$$

The range of is 0 to 1. Higher values are preferred for better clustering, the best possible value is 1.

5. *F-score:* F-score F_β is a combination of precision P and recall R; it measures the extent to which a cluster contains only objects of a particular class and all objects of that class:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{2.14}$$

The parameter β is the weighting factor for the importance of the recall (or precision). In SRC domain, we give more weight to recall (β must not be zero, recall weight increases as β increases). The range of F_β is 0 to 1. Higher values are preferred for better clustering, the best possible value is 1.

Similarity Oriented measures

These approaches measure the extent to which two objects that are in the same class are in the same cluster and vice versa [126]. The following subsections use the notations: TP is number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

1. *Rand Index:* Rand index [112] is a measure of the percentage of correct decisions made by the algorithm. It is computed by the formula:

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.15}$$

False positives and false negatives are equally weighted in Rand Index, this may not be suitable for some clustering applications. The value of Rand Index is dominated by true negatives in case of SRC. The range of RI is 0 to 1. Higher values are preferred for better clustering, the best possible value is 1.

2. *Jaccard Coefficient:* The Jaccard Coefficient [126] is used to quantify the similarity between the ground truth and the clusters. It takes a value between 0 and 1, 1 means that the two datasets are identical, and 0 indicates that the datasets have no common elements. Hence the best value is 1. It is defined by the following formula:

$$J = \frac{TP}{TP + FP + FN} \tag{2.16}$$

2.6 Text Mining

The text mining techniques needed for processing the search results are described in the following sections. Usually the search result's title and snippet form a document for processing the search result.

2.6.1 Tokenization

The title and snippet of each search result need to be split into independent units called tokens. Usually white space characters are used as separators for the tokens. The tokenizer must be able to handle special symbols, URLs and filenames.

2.6.2 Stopword removal

Words like "a", "for", "on","the", etc are frequent and unimportant words which do not carry much meaning. They must be ignored. SMART [60] is a stop-word list that can be taken as a guideline. Stop-words may vary per document set. For example, "database systems" could be an important keyword in a newspaper. However, it may be considered as a stop-word in a set of research papers presented in a database systems conference [53]. Usually search results contain

special characters like ellipsis ("...") and "amp", which need to be taken into account.

2.6.3 Stemming

English words like "look" can be inflected with a morphological suffix to produce "looks, looking, looked". They share the stem word "look". The aim of stemming is to remove the inflectional prefixes and suffixes of each word and to reduce to the stem word. The stem word may not always be a dictionary term. Porter Stemmer [104] is the commonly used stemming algorithm.

2.6.4 Vector Space Model

The vector space model is the the representation used in text mining and Information Retrieval. Each document is represented as a vector v in the t dimensional space R^t, given a set of d documents with a set of t terms. Each search result is treated as a document, represented as a vector of weights formed by Term Frequency and Inverse Document Frequency (TF-IDF). TF-IDF reflects how important a word is to a document in a collection. Term Frequency is the number of occurrences of term t in document d, it is denoted as $tf_{t,d}$. Inverse Document Frequency diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. When the total number of documents in a collection is N, document frequency df_t is the number of documents in the collection that contain a term t, then Inverse Document Frequency $idf_t = \log \frac{N}{df_t}$. The weight of each term is $tf\text{-}idf_{t,d}$.

$$tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t \tag{2.17}$$

Each search result document is represented as a vector of $tf\text{-}idf$ weights. Using $tf\text{-}idf$ weights, each document is represented as a vector. All these vectors form a term-document matrix to enable application of data mining tools.

2.6.5 Feature Extraction

Features are the atomic entities by which each object is represented. The simplest and commonly used features are words. Other features used are n-grams, frequent phrases etc. Many features are irrelevant for the task of assessing similarity or dissimilarity between documents, and can be discarded. The feature extraction techniques are available in [142] and [79].

2.6.6 Cosine similarity

Cosine similarity measures the similarity of two documents when each document is represented as a vector. It is based on the observation that when two vectors have approximately the same features then they should "point" in similar direction in the space determined by the term-document matrix, regardless of their Euclidean distance. To calculate similarity between two documents, the cosine of the angle between them is used, by calculating the dot product between their document vectors.

Hence cosine similarity between vector representation of documents d_i and d_j in the term vector space is defined as:

$$similarity(d_i, d_j) = cos(\theta) = \frac{d_i \cdot d_j}{|d_i||d_j|} \tag{2.18}$$

CHAPTER 3

Hybrid Classifier Based on Inferred Attribute

Dimensionality reduction is used during preprocessing phase to avoid the effects of curse of dimensionality. Dimensionality reduction which yields succinct representation of original knowledge enables building classifiers (in fact any data mining tools) that are useful in applications which require quick response times. This chapter details an effective dimensionality reduction approach induced by the knowledge acquired through inferred attributes. The proposed method is demonstrated with empirical analysis using decision tree as classifier.

3.1 Introduction

Classification is the task of learning a **target function** f that maps each conditional attribute set $\mathbf{x}$ to one of predefined classes y, based on a labeled training set of objects, to estimate the label of unlabeled objects. The decision attribute in a classification system is expected to be dependent on conditional attributes. The degree of dependency and the efficiency of the tools adopted will qualify the resultant classifier. These dependencies, in general, are highly non linear as well as implicit. Thus the problem of classification has become a challenging problem for the research groups of data mining, machine learning, etc. Several approaches like k-NN, Neural Networks, Discriminant analysis, Decision Trees and Rule based approaches are developed to classify new data object for its decision label. Hybrid decision trees are developed by [89], [111] and [38] using the rough set theory [101]. Dimensionality reduction achieved by rough set theory can be attributed for the performance gain of the above hybridizations.

The real world problems can be handled in several ways. Researchers have brought out techniques by formulating and solving the problem in a derived space rather than original space. Some variants of Neural Networks such as RBF Networks, SOMs and non linear SVMs etc are some of the examples of this philosophy [17] . The same philosophy with a supervised approach has been adopted by [130] and [131] in system designing, to handle the uncertainty issues. They also demonstrated a novel approach for constructing such transformations which are referred

to as intermediate low fidelity (ILF). The ILF mimics the response of the system indicating that ILF possesses adequate knowledge.

Further, oblique decision trees [55] are compact and more accurate. While most of the decision tree inducing algorithms create tests at each node that involve a single attribute of the data, oblique decision trees can use multiple attributes. Oblique decision trees use multivariate tests that are not necessarily parallel to an axis. However, learning the appropriate test condition at each node is a computationally challenging task. Constructive induction [88] is another approach to improve expressiveness of a decision tree. This approach creates compound features from original attributes to create complex splitting functions.

For many practical tasks, the trees produced by tree-generation algorithms are not comprehensible to users due to their size and complexity [15]. The number of classification rules depends on the breadth of the decision tree, where as the length of the classification rules depends on the depth of the decision tree. The classification complexity of an unknown object is proportional to the depth of the tree. So developing compact rules without compromising the accuracy of the classifier is the interest of researchers [89].

These observations is the inspiration to develop derived space (in explicit forms) and to study the impact of this derived information on classification.

The essence of the attributes which preserves the information and meaning will enable compact representation of the decision system. The attributes of the derived space are referred hereafter as inferred attributes(IA). IA encapsulates the potential information required for predicting the decision characteristics. Appending inferred attributes and the corresponding values to the given decision table as conditional attributes will be referred to as Augmented Decision Table (ADT). ADT is in higher dimensions due to dimensionality expansion. Some of the attributes become redundant in the presence of IA. Thus the dimensionality enhancement enables significant dimensionality reduction. The dimensions are reduced by finding the *reduct* [27] of ADT. *Reduct* is a minimal knowledge representation scheme developed in [101]. It is a subset of attributes which can, by itself, fully characterize the knowledge in the database/decision system.

The proposed system is a three step hybrid version of inferred attribute induction, *reduct* computation and classifier construction. The system is demonstrated with Linear Regression and Quadratic Regression for IA, Quick Reduct algorithm [27] for *reduct* computation and CART [14] for the classifier.

3.2 Inferred Attribute

Let $C = \{a_1, a_2, ..., a_n\}$ be the set of conditional attributes and $D = \{d\}$ be the set of decision attribute of the decision system $< U, C \cup D >$. A function which maps space of C to space of D qualifies to be called as an approximation to d. This approximation has potential to inherit characteristics of C and mimicking D. This approximation is called as Inferred Attribute and is labeled as IA in the current study. This IA (functional forms) can be considered as explicit as possible with a reasonable evaluation time. The parameters of the functional form of the IA can be fixed optimally by minimizing the departure from d. To start with, the well known statistical regression approaches [12, 139] for constructing regression models for a given independent variables $X = (X_1, X_2, ..., X_n)$ and the dependent variable Y, as $Y_{estimate}$. $Y_{estimate}$ can be considered as a methodology for building IAs. In a decision system, C is X and D is Y. IA is an expression (model form) in C which may have some unknowns, known as parameters B. One can get an estimate of B based on training data set. As pointed out in Introduction, the current study confines to two models i.e., Linear Regression and Quadratic Regression.

3.2.1 Algorithm ComputeIA

Algorithm for finding the Inferred Attribute as IA and model parameters based on mean error sum of squares is described in Algorithm 1. It takes training set and returns estimated parameters for the model. And also some statistical measures for assessing the IA quality are produced by this algorithm.

3.2.2 Illustration of IA

A sample from Iris dataset [129] is selected for demonstrative purpose. It is divided into two parts, training data and test data. The model forms for Linear regression(IA1), Quadratic regression(IA2) are determined based on training data and evaluated for both training data and test data. The Table 3.1 shows the training data and the inferred attributes obtained using the Algorithm 1 (The columns headed by IA1 and IA2 are the inferred attributes obtained for the training data formed with the columns 1,2,3,4 and 7). The coefficients obtained for the linear regression function to derive IA1 are
w0 = 2.6428, w1 = -0.5469, w2 = 0.0409, w3 = 0.5468 and w4 = 0.3157.

The coefficients obtained for the quadratic regression function to derive IA2

Algorithm 1: ComputeIA

Input : DT = [X Y] is the training data with condition attributes in X and decision attribute Y, f($X \mid \theta$) is the model form

Output: Estimates of parameters of the model θ_{Est},IA,ESS,TSS,R^2, estV

begin

// the least square estimate of the parameters θ

$\theta_{\mathsf{Est}} \leftarrow$ arg(min(Error sum of squares));

// evaluate f(X $\mid \theta_{\mathsf{Est}}$) for the given X

IA $\leftarrow$ f(X $\mid \theta_{\mathsf{Est}}$) ;

// compute Error Sum of Squares

ESS $\leftarrow$ $(\mathsf{Y} - \mathsf{IA})^T(\mathsf{Y} - \mathsf{IA})$;

// compute total sum of squares

TSS $\leftarrow$ sum of squares of deviates of Y from its mean;

// compute coefficient of determination

$\mathsf{R}^2 \leftarrow (\mathsf{TSS} - \mathsf{ESS})/\mathsf{TSS}$;

// compute estimation of Variance

// N is the size of training data

// k is the number of parameters to estimate

estV $\leftarrow$ ESS/(N − k);

end

are
w0 = 6.0021, w1 = -1.3787, w2 = -2.0224, w3 = 2.3022 and w4 = -2.5504,
w5 = -1.2227, w6 = 2.9467, w7 = 1.8089, w8 = -0.3804 and w9 = -1.2216,
w10 = -2.6044, w11 = 1.7107, w12 = 0.1310, w13 = -4.5856 and w14 = 7.2590.

These coefficients are used to form the linear and quadratic regression functions to obtain IA1 and IA2 values for the training data as well as test data. The test data and its IAs are shown in Table 3.2.

3.2.3 Explicit and Implicit IA

Explicit functions express the decision attribute as a function of conditional attributes. Linear Regression and Quadratic Regression are explicit functions demonstrated in this thesis. Similarly any suitable explicit function which can mimic the decision attribute can be used by incorporating the domain knowledge.

A function of conditional attributes is constructed to propose an estimate of D. Depending on the domain knowledge this function will be a good estimate for D. One can consider the features based on the Singular Value Decomposition of

Table 3.1: Training data and the Inferred Attributes

Sepal length	Sepal width	Petal length	Petal width	IA1	IA2	Class
6.1	3	4.6	1.4	2.39	2.20	2
7.7	2.6	6.9	2.3	3.04	3.13	3
6.2	2.8	4.8	1.8	2.56	2.95	3
5	3.5	1.6	0.6	1.12	1.10	1
5.1	3.8	1.5	0.3	0.92	0.88	1
6.6	3	4.4	1.4	2.00	1.82	2
4.9	3.1	1.5	0.1	0.94	1.05	1
6	2.2	5	1.5	2.66	2.99	3
6.9	3.1	4.9	1.5	2.15	2.14	2
6.5	2.8	4.6	1.5	2.19	2.02	2
4.6	3.2	1.4	0.2	1.09	0.88	1
5	2.3	3.3	1	2.12	2.03	2
5.8	2.6	4	1.2	2.14	2.05	2
6.4	2.9	4.3	1.3	2.02	1.88	2
6.1	2.9	4.7	1.4	2.44	2.34	2
7.6	3	6.6	2.1	2.88	2.90	3
6.8	2.8	4.8	1.4	2.10	1.99	2
6.3	2.8	5.1	1.5	2.57	2.62	3
5.8	2.7	5.1	1.9	2.97	2.96	3
5.4	3.7	1.5	0.1	0.69	1.07	1

associated conditional attributes as IA.

3.3 Hybrid Classifier

IA inherits the characteristics of C and mimics D. Hence, it is expected that a better performance can be achieved by augmenting IA to the original decision system. This augmentation increases the dimensionality, and some of the attributes of C may become redundant in presence of IA. Thus it is worth reducing the dimensions. Studies [38, 89, 111] demonstrated effectiveness of dimensionality reduction and constructing improved decision trees by hybridizing rough set theory in decision trees. The essence of these works is to replace independent attributes in decision system by a *reduct*.

Each of the IAs is considered as additional condition attribute. New decision tables are constructed and considered for developing hybrid classifier. In the current study IA is based on conditional and decision attributes of the training data which can be used for predicting decision variable thus acts as an intermediate fidelity. IA has an explicit expression in terms of conditional attributes. Augmented Decision Table (ADT) is obtained by appending IAs to the conditional

Table 3.2: Test data and its Inferred Attributes

Sepal length	Sepal width	Petal length	Petal width	IA1	IA2
5.6	2.7	4.2	1.3	2.40	2.27
6.2	2.2	4.5	1.5	2.28	1.64
4.6	3.4	1.4	0.3	1.13	0.79
5.5	2.5	4	1.3	2.33	2.26
6.5	3.2	5.1	2	2.64	3.47
6.7	2.5	5.8	1.8	2.82	3.00
5.1	3.7	1.5	0.4	0.95	0.84
5.5	2.6	4.4	1.2	2.53	2.71
5.7	2.9	4.2	1.3	2.35	2.10
6.3	2.5	4.9	1.5	2.45	2.49

attributes. A *reduct* [27] is obtained for ADT and Reduced ADT is obtained by taking the projection of ADT with respect to this *reduct*. Reduced ADT is used for the purpose of constructing classifier. It is proposed to bring out the potential characteristics of inferred attribute through statistical investigations as well as the index of knowledge representation in Rough sets, kappa. CART has been used in the present study (any other classifier can be used) to demonstrate the role of IA in classification. As the inferred attribute is an explicit expression, thus, the rules derived from the decision tree constructed based on Reduced ADT are explicit and interpretable.

3.3.1 Augmented Decision Table

Augmented decision tables for training and test data are obtained by appending the IA(s) to the conditional attributes. Hence the ADT for training data contains

$$< \text{conditional attributes, inferred attribute(s), decision attribute} > \quad (3.1)$$

the ADT for test data contains

$$< \text{conditional attributes, inferred attribute(s)} > \quad (3.2)$$

The Tables 3.1 and 3.2 are the ADTs of training data and test data respectively.

3.3.2 Reduced Augmented Decision Table

Minimal representation of decision system using concept approximation approach results in removal of dispensable conditional attributes and retains only indispensable. These indispensable attributes constitute a *reduct*, which is a knowledge representation. The Tables 3.3 and 3.5 are the reduced decision tables of the training data and the test data respectively. For the demonstrative data of Iris, when the inferred attribute based on linear regression and quadratic regression are used as the inferred attributes individually, they alone became the conditional attributes by using the *reduct*.

IA1	Class
2.39	2
3.04	3
2.56	3
1.12	1
0.92	1
2.00	2
0.94	1
2.66	3
2.15	2
2.19	2
1.09	1
2.12	2
2.14	2
2.02	2
2.44	2
2.88	3
2.1	2
2.57	3
2.97	3
0.69	1

Table 3.3: RDT of the training data using linear regression

IA2	Class
2.20	2
3.13	3
2.95	3
1.10	1
0.88	1
1.82	2
1.05	1
2.99	3
2.14	2
2.02	2
0.88	1
2.03	2
2.05	2
1.88	2
2.34	2
2.90	3
1.99	2
2.62	3
2.96	3
1.07	1

Table 3.4: RDT of the training data using quadratic regression

IA1
2.40
2.28
1.13
2.33
2.64
2.82
0.95
2.53
2.35
2.45

Table 3.5: RDT of the test data using linear regression

IA2
2.27
1.64
0.79
2.26
3.47
3.00
0.84
2.71
2.10
2.49

Table 3.6: RDT of the test data using quadratic regression

3.3.3 Algorithm for Building Hybrid Classifier

The pseudo code of the proposed hybrid classifier is presented in Algorithm 2. The training data is fed to the Algorithm 1 to find the coefficients of the function to arrive at the inferred attribute. The ADT is formed using the IAs. The minimal set of conditional attributes is arrived by calling the function "findReduct". The RDT is arrived by considering the *reduct* attributes. RDT is used as the training data to build the classifier using the "buildClassifier".

Algorithm 2: Hybrid Classifier

Input : DT = [X Y] is the decision table with condition attributes in X and decision attribute Y, $f(\mathsf{X}|\,\theta)$ is the model form.

Output: Classifier

begin

```
// find the estimate of Y using DT and model form f(X| θ)
[IA, θ_Est] ← computeIA (DT, f(X| θ)) ;
// Augment the conditional attributes with IA
ADT ← [X IA Y] ;
ReductA ← findReduct (ADT) ;
XR ← attributes corresponding to ReductA of ADT;
// reduced decision table contains ReductA attributes and Y
RDT ← [XR Y] ;
// build classifier on reduced decision table
Classifier ← buildClassifier (RDT) ;
```

end

3.3.4 Algorithm for Classification with Hybrid Classifier

The IA function(s) arrived at the training phase of the classifier is used to compute the IA(s) for the test data. The augmented decision table for the test data is formed by using the IAs obtained on the test data using the IA function(s) arrived from the training phase. The RDT is formed by considering only the attributes available in the *reduct*, which is computed at the training phase of the classifier. The classifier is built using Algorithm 2 during the training phase of the classifier and used to find the class of each of the test data object using the function "Classify".

Algorithm 3: Classification with Hybrid Classifier

Input : ReductA, `Classifier`, Xtest, $f(X \mid \theta_{Est})$

`// Xtest is the test data objects to classify`

`// ReductA,` $f(X \mid \theta_{Est})$ `are arrived in training phase`

Output: Class

begin

IAtest $\leftarrow f(\text{Xtest} \mid \theta_{Est})$;

ADTtest $\leftarrow$ [Xtest IAtest];

XRtest $\leftarrow$ attributes corresponding to ReductA of ADTtest;

Class $\leftarrow$ `Classify` (`Classifier`, XRtest);

end

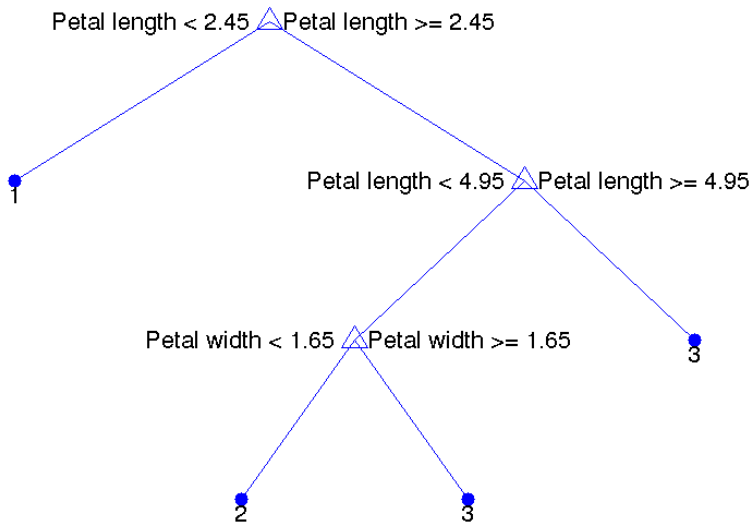

Figure 3.1: Decision tree on the Iris demo data

3.3.5 Illustration of the Hybrid classifier using CART

The Figure 3.1 shows the decision tree for the demonstrative data of Iris using CART [14]. The decision trees obtained with the reduced decision Tables 3.5 and 3.4 are shown in Figure 3.2 and Figure 3.3. These figures are obtained using the Matlab. By considering the total Iris data, it is observed that the depths of the CART trees are 4, 4, 2 and 4 (with number of leaf nodes as 5,5,3 and 5), for the original Iris data, for confining to a *reduct* of Iris, *reduct* of ADT with linear regression and ADT with quadratic regression respectively. The respective impurities at the leaves are 0.3551, 0.3551, 0.0566 and 0. This shows that the impurity has a considerable reduction with ADTs. However, this is possible with an IA which captures good amount of information.

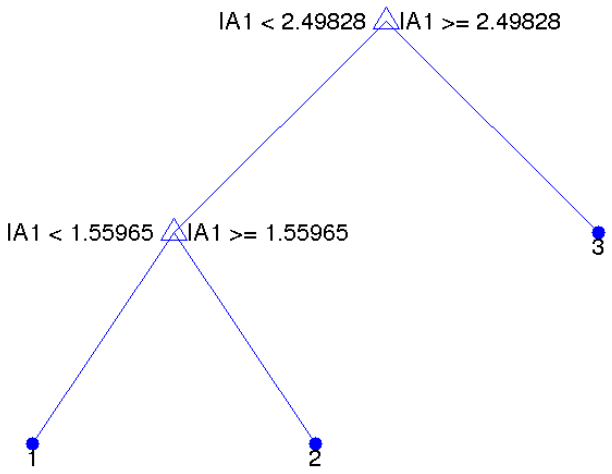

Figure 3.2: Decision tree on the RDT with linear regression of Iris demo data

3.4 Experiments and Results

The effectiveness of the hybrid classifier is demonstrated and validated using numeric and text datasets. The numeric datasets are used to demonstrate that the tool can serve as a standalone Data Mining tool, the text datasets are for the tool's suitability for SRC.

3.4.1 Numeric Data

The experiments are conducted using standard datasets available in UCI machine learning repository [129] in Matlab environment and RSES [7, 113]. The IAs are obtained by using IA1 and IA2. Statistical analysis has been adapted for assessing the mimicking properties of IA. The Rough Set Exploration System (RSES) is used for obtaining the *reduct* of original decision table and ADTs. For comparative study, RDTs are constructed using respective reducts. Decision tree using CART is constructed for original as well as Reduced Decision tables. Experiments are conducted using tenfold cross validation on several benchmark data sets from UCI machine learning repository.

Table 3.7 summarizes the characteristics of datasets and the reducts obtained for Original DT and ADT with IA1 and ADT with IA2. The significant impact of the IA is seen in the reduction of length of the *reduct* (see Figure 3.4). For

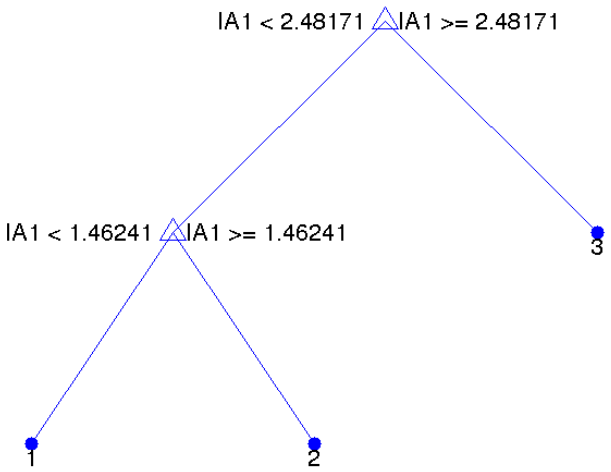

Figure 3.3: Decision tree on the RDT with quadratic regression of Iris demo data

Table 3.7: Reduct Information

Data set Name	#Objects	# Attributes	Kappa	Reduct of DT	Reduct of ADT with IA1	Reduct of ADT with IA2
Iris	150	4	1	{1,2,3,4}	{IA1}	{IA2}
Wine	178	13	1	{1,2,4,5,8,12,13}	{IA1}	{IA2}
WDBC	699	10	1	{1,2,5,6,7}	{IA1}	{IA2}
Blood Transfusion	748	4	0.733	{1,2,4}	{2,IA1}	{1,IA2}
Abalone	4177	8	1	{1,2,4,5,6,7,8}	{7,8,IA1}	{2,4,IA2}
Ecoli	336	7	1	{1,2,5,6,7}	{1,5,IA1}	{5,6,IA2}
Yeast	1484	8	1	{1,2,3,4,7}	{1,4,IA1}	{3,4,7,IA2}
Page-blocks	5473	10	0.992	{1,2,3,4,5,6,7,8}	{4,5,IA1}	{4,8,IA2}
Wine Quality - Red	1599	11	1	{1,3,4,5,7}	{4,7,IA1}	{4,10,IA2}
Pima-Indians Diabetes	768	8	1	{1,2,3,7}	{3,IA1}	{1,IA2}

datasets like Iris, Wine and WDBC, IA alone became the *reduct*.

Table 3.8 presents the model characteristics such as error sum of squares (ESS), Coefficient of determination (R^2) and estimate of the error variance (estV) with IA1 and IA2 respectively. Table 3.9 summarizes the results of classification accuracy of CART for DT (Original table), RDT (Reduced table of original DT), RDT-IA1 (Reduced table of ADT with IA1), RDT-IA2 (Reduced table of ADT with IA2). It also summarizes the total time taken in each of these models which includes building CART and testing in ten folds.

For instance, considering the Abalone data set (one among the datasets with least performance), which has 4177 objects with 8 conditional attributes and 1 decision attribute, to observe the impact of IA. By adopting RSES the selected shortest length reducts are {1,2,4,5,6,7,8}, {7,8,IA1} and {2,4,IA2} for original

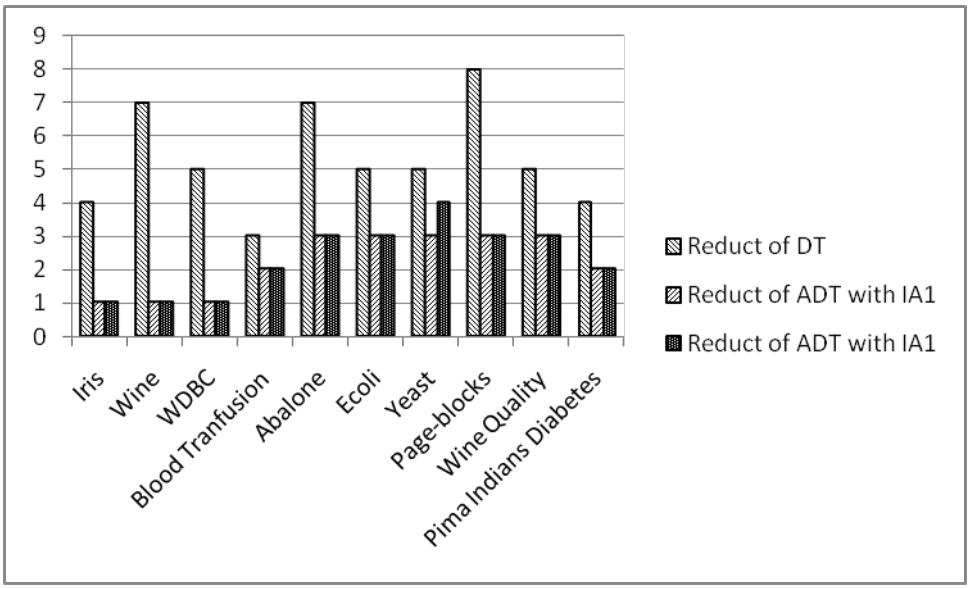

Figure 3.4: Reduct Length Comparisons with and without IA

Table 3.8: Inferred Attribute Model Characteristics

Data set name	IA1			IA2		
	ess	R^2	estV	ess	R^2	estV
Iris	6.97	0.93	0.05	4.91	0.95	0.04
Wine	10.62	0.9	0.06	1.72	0.98	0.02
WDBC	25.61	0.84	0.04	15.83	0.9	0.03
Blood Transfusion	118.14	0.13	0.16	112.24	0.17	0.15
Abalone	680.66	0.34	0.16	632.82	0.38	0.15
Ecoli	862.43	0.63	2.63	650.37	0.72	2.17
Yeast	12566.07	0.08	8.52	12015.89	0.12	8.35
Page-blocks	1987.14	0.3	0.36	1306.77	0.54	0.24
Wine Quality - Red	666.41	0.36	0.42	589.17	0.43	0.39
Pima-Indians Diabetes	121.57	0.3	0.16	108.03	0.38	0.15

DT and for ADT with IA1 and ADT with IA2 respectively with kappa as 1. It is also observed that for ADT all the possible shortest reducts contain IA not only for Abalone but also for all the considered benchmark data sets. The classification performance of Abalone using ten-fold cross validation test by adapting CART are 70.70, 71.34, 70.31 and 73.40 for Original DT, RDT, RDT-IA1 and RDT-IA2 respectively (See Table 3.9 and Figure 3.5). These results indicate a marginal gain in accuracy due to IA. The computational time for tenfold test for each of these tables are 611.07 (549.16 building time+82.17 for testing time), 558.92 (497.29+83.82), 375.79 (281.75+99.48) and 299.32 (246.84+59.01) which indicates the enhancement achieved due to the inclusion of IA which is shown in Figure 3.6 and tabulated in Table 3.9

By performing paired t-test it is observed that, the building time of CART is

Table 3.9: Performance comparison for UCI datasets

Data set name	Accuracy (%)				Time (milliseconds)			
	DT	RDT	RDT-IA1	RDT-IA2	DT	RDT	RDT-IA1	RDT-IA2
Iris	93.33	93.33	98	98	44.59	10.63	6.59	5.14
Wine	88.68	92.08	97.78	97.78	20.33	16.62	5.18	5.12
WDBC	94.57	94.71	96.71	96.71	38.1	24.12	5.35	4.86
Blood Transfusion	75.13	75.13	76.61	75.81	76.89	65.11	19.9	18.32
Abalone	70.7	71.34	70.31	73.4	611.07	558.92	375.79	299.32
Ecoli	79.76	80.35	81.65	81.24	38.42	34.1	32.39	26.53
Yeast	53.24	51.42	50.54	50.61	289.4	253.88	176.5	207.36
Page-blocks	96.42	96.53	95.5	95.85	403.71	341.41	177.97	179.85
Wine Quality	60.54	52.47	55.66	54.6	396.63	256.41	170.99	168.68
Pima-Indians Diabetes	70.83	67.07	76.69	75.91	110.12	85.61	28.97	18.39

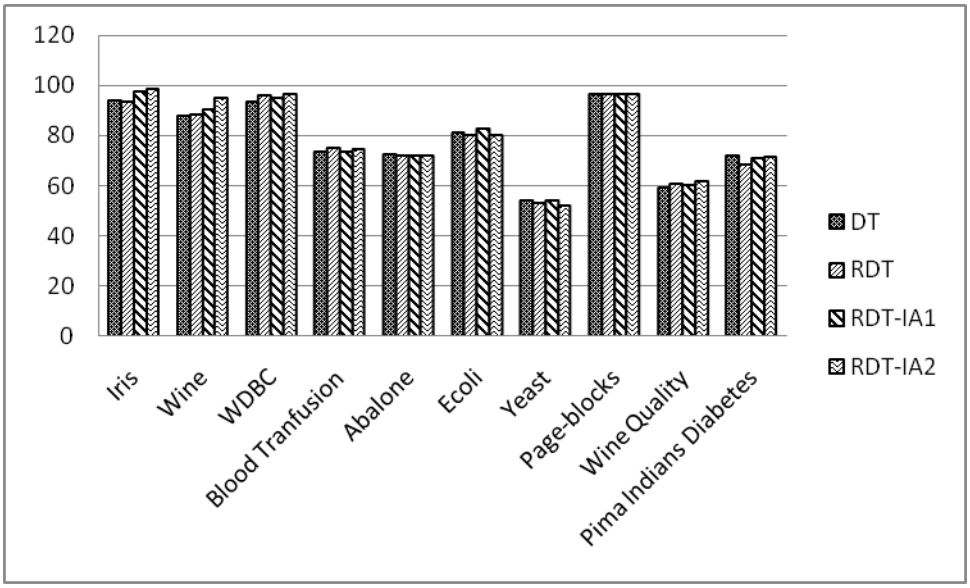

Figure 3.5: Comparison of classification accuracies in percentages

statistically significantly low when the data for training is used with Inferred Attribute, indicating that IA plays a significant role in the learning phase of decision tree. But the testing time of CART is statistically significantly low only for IA2 (Quadratic Regression). One can observe that the reasonable IA will reinforce learning but apt IA only will reinforce testing.

3.4.2 Text Data

The "4 Universities Data Set" [132] is used to illustrate the Hybrid classifier on text data. This data set contains WWW-pages collected from computer science departments of various universities in January 1997 by the World Wide Knowledge Base (WebKB) [29] project of the CMU text learning group [28]. The 8,282 pages in this dataset were manually classified into seven classes: student, faculty, staff, department, course, project, and other. The distribution of the classes is presented

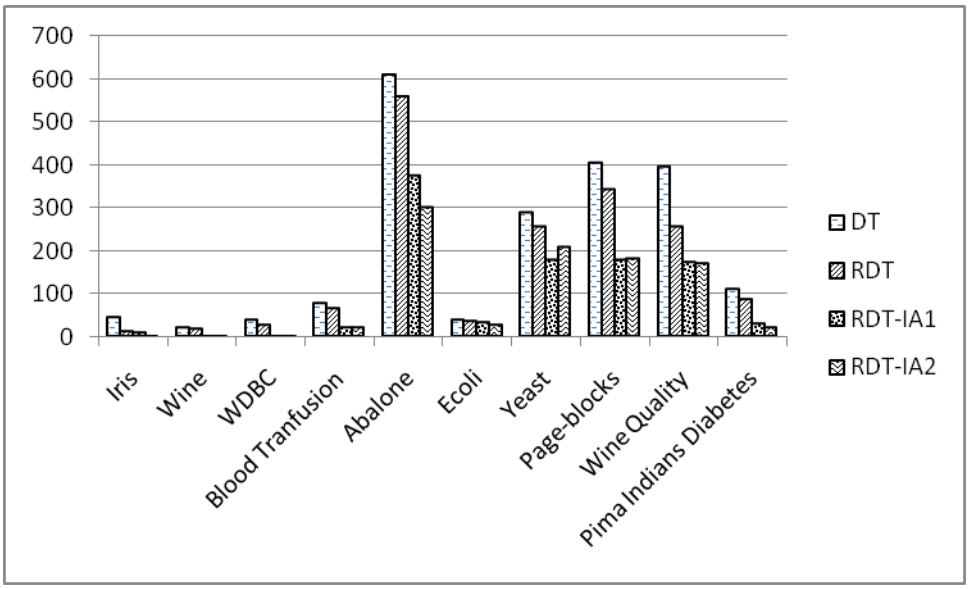

Figure 3.6: Comparison of time taken for Classifiers in milliseconds

Table 3.10: The 4 Universities data set class distribution

Class	# Web pages
student	1641
faculty	1124
staff	137
department	182
course	930
project	504
other	3764

in Table 3.10. The pages which may not be judged as "main page" are given under the class 'other'. For example, only the home page of faculty member is given under the class 'faculty', while the curriculum vitae, publication and research interest pages come under the class 'other'. 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.

The web pages are preprocessed according to the techniques presented in section 2.6, each web page is represented as a vector of TF-IDF values. The classes

Table 3.11: Performance comparison for 4 Universities data set

Classification Method	Accuracy
kNN (k = 10)	0.7256
Centroid (Normalized Sum)	0.8266
Naive Bayes	0.8352
SVM (Linear Kernel)	0.8582
Hybrid Classifier	0.8342

'staff' and 'department' contain few web pages when compared to other classes, 'other' class web pages deviate much from the remaining classes. Hence these three classes' pages are removed and the data set is used with the four classes 'student', 'faculty', 'course' and 'project'. The data set is randomly divided such that two thirds of it forms training data and the remaining one third forms test data. The TF values are normalized with respect to the web page size. The IDF values obtained on the training data are used while forming the TF-IDF vectors of text data. This facilitates the classifier model to be suitable for test data.

The decision table for the classifier consists of the TF-IDF vectors. Inferred Attribute is computed using Linear Regression on the TF-IDF vectors, and augmented to the decision table. RDT is obtained with the usage of QuickReduct algorithm [27]. The original decision table contains 7568 attributes and the reduced decision table contains 948 attributes. CART [14] is used as the classifier on this reduced decision table. The results of the classifier are reported in Table 3.11 in comparison to the popular classification methods accuracies taken from [18]. The accuracy of the Hybrid classifier is on par with the best classifiers shown in Table 3.11. Hence the proposed classifier is useful for classifying the web pages and search results obtained from search engines. The proposed classifier is used in the Meta Search Engine to classify the search results.

3.4.3 Observations

1. From Table 3.1 and Figure 3.4, it is observed that the IA is mimicking the decision attribute and the dimensionality reduction of ADT is significantly smaller than that achieved by *reduct* on the original data.

2. From Tables 3.8 and 3.9, the accuracy of the classifier improved with IA. The accuracy is much better whenever R^2 is high. This observation suggests that the model used for IA must be apt to capture significant knowledge. This demands domain knowledge of the system.

3. Significant reduction in the number of conditional attributes makes the tree

depth to be smaller, in turn producing rules of shorter length. Hence, the knowledge acquired through IA is expected to have a quality enhancement in the performance of the knowledge engine (classifier).

3.5 Conclusions

A hybrid classification method developed in this chapter produces shorter rules without compromising the accuracy. This methodology performs a meaningful dimensionality enhancement (with IA) which in turn yielded better dimensionality reduction. Dimensionality enhancement is demonstrated with regression models and classification is demonstrated with CART. The proposed hybrid classifier has been validated empirically on various numerical data sets as well as text data. IA is present in all the minimal length reducts and has significant influence on the *reduct* length. Time required for classification using this model turned out to be significantly low. The demonstration on WebKB data set is encouraging to embed this methodology in search engines.

Appropriate IA with non-linear forms will be in a position to address the problems like XOR that are hard to be modeled by decision trees.

CHAPTER 4

Coefficient of Variation based Decision Tree (CvDT)

Decision trees are popularly used for classification. The efficiency of a decision tree is guided by the attribute selection measure used for internal node splitting. An attribute selection measure called CvGain is proposed which is suitable for the environments that demand on the fly classification. This chapter presents the proposed attribute selection measure and its salient features. The applicability of the decision tree constructed using CvGain for dynamic environments like Web is demonstrated through empirical analysis. The methodology for using the proposed decision tree in distributed environments is detailed. The suitability of the same for continuous decision attribute is demonstrated.

4.1 Introduction

Decision trees are well known for classification [53] as they are easy to interpret and simplify the complex decision making process. ID3 [107], C4.5 [110], CART [14] are few popular implementations of decision trees. ID3 algorithm is the first decision tree implementation. Building a decision tree follows a greedy approach for choosing the best attribute for splitting at a node. Splitting criteria plays a vital role in building a decision tree. Information Gain, Gain ratio, Gini index, Chi square statistics, Variance, Standard Deviation and Kappa index are the well known splitting criteria. Coefficient of Variation (Cv) [119], which is a measure of consistency of data is used in applied domain. It is free from units and indicates relative standard deviation. Hence, it imitates tolerance behavior of human. The application of Cv for constructing risk trees in managerial studies is demonstrated in [34]. Cv has not attracted the researchers of data mining as a splitting criteria till date.

Minz and Jain [89] has applied ID3 on the reduced data obtained by reduct attributes based on Rough set theory [101]. Reduct selects only predominant attributes and leads to dimension reduction without loss of information. Hence, the resulting tree generated fewer classification rules with comparable classification accuracy to ID3. A *reduct* based decision tree where the splitting attributes are

selected according to their order of presence in the reduct is proposed by [111]. To the best of our knowledge, the latest splitting criteria used is Gainfix, which is proposed in [38].

Since its inception, ID3 has been thoroughly studied by various researchers. ID3 uses Information Gain as the splitting criteria. But Information Gain uses frequencies ignoring its ordinates and is based on the Entropy which invokes logarithmic function several times. Decision trees based on standard deviation and variance considers the measurement values also besides frequency. However, an attribute with higher variance (SD) will be preferred ignoring the location of the distribution. These trees tend to be overfit/underfit.

The computation of Cv is less expensive as it uses simple arithmetic operations and square root function. And standard deviation is normalized w.r.t mean and it is a measure of consistency. The similarity between coefficient of variation and Gini index brought out by [51], inspired in inventing a coefficient of variation based decision tree, in short CvDT.

4.2 Attribute Selection Criteria

ID3 [107] proposed by Quinlan uses Information gain for attribute selection, which is based on information theory. But Information Gain is biased towards multi-valued attributes.

C4.5 [110] is a successor of ID3 and uses Gain ratio, which is an extension of Information gain. Gain ratio overcomes the biasing for multi-valued attributes by applying some normalization to Information Gain. But Gain ratio tends to prefer unbalanced splits.

The Gini index which considers a binary split for each attribute is used by CART [14]. But Gini index also prefers multi-valued attributes and has a difficulty in dealing large number of classes.

These approaches are frequency centric ignoring magnitude. The limitations of impurity based measures like Information gain and Gini index are given by [43].

Standard deviation is used as a measure of error in M5 trees [108]. The splitting criteria is guided by the maximum error reduction.

The Gainfix measure developed based on Rough set theory is based on Kappa index. It is used in FID3 [38]. It considers relationship between condition attributes and decision attributes in addition to Information Gain.

Table 4.1: GPA Data

Student	A_1	A_2	D
S1	3	2	2
S2	3	1	3
S3	4	3	1
S4	2	1	3
S5	3	3	2

Cv being a measure consistency of data, which is free from units and which indicates relative standard deviation motivated to propose *CvGain*, an attribute selection measure based on *Cv*.

4.3 CvGain

The similarity between coefficient of variation and Gini index through Lorenz function is brought out by [51]. Theoretical aspects for considering the square of the coefficient of variation as an equality measure in the same way as is the Gini index is provided by them.

Adaptive discrimination nature based on relative magnitude can be realized through coefficient of variation (Cv), which is not possible through standard deviation or variance. Further Cv addresses consistency aspects of the data. Hence, the attribute selection measure called *CvGain* is proposed based on Cv.

4.3.1 Coefficient of Variation

Coefficient of Variation [12, 119] is the ratio of standard deviation σ and mean μ.

$$Cv = \sigma/\mu \times 100 \tag{4.1}$$

Coefficient of Variation is a dimensionless number and hence it is suitable for comparing data in different units or with widely different means. Cv is defined for non zero mean. The computation of Cv is illustrated with Table 4.1 data. This data contains two attributes: High School GPA (called as A_1) and College GPA (called as A_2).

The computations of Cv for each attribute of GPA data are given below.

$$Cv(A_1) = \sigma(A_1)/\mu(A_1) * 100 = (0.6325/3) * 100 = 21.0819 \tag{4.2}$$

Table 4.2: Conditional table with $A_1 = 2$

Student	A_1	D
S4	2	3

Table 4.3: Conditional table with $A_1 = 3$

Student	A_1	D
S1	3	2
S2	3	3
S5	3	2

$$Cv(A_2) = \sigma(A_2)/\mu(A_2) * 100 = (0.8944/2) * 100 = 44.7214 \tag{4.3}$$

$$Cv(D) = \sigma(D)/\mu(D) * 100 = (0.7483/2.2) * 100 = 34.0151 \tag{4.4}$$

4.3.2 CvGain Computation

Let DT be the decision table which is preprocessed such that Cv can be computed. The decision table with conditional attributes $A_1, A_2, \ldots, A_n$ and the decision attribute D is represented as $DT = [A_1, A_2, \ldots, A_n, D]$. The Coefficient of Variation of decision attribute D is given by

$$Cv(D) = \frac{\sigma(D)}{\mu(D)} \times 100 \tag{4.5}$$

Coefficient of Variation of D conditioned on A_i having v distinct values $(a_1, a_2, \ldots, a_v)$ is given by:

$$Cv(D \mid A_i) = \sum_{j=1}^{v} P_j \; Cv(D \mid A_i = a_j) \tag{4.6}$$

Where a_j is the j^th possible value of A_i with chance P_j. And

$$CvGain(A_i) = Cv(D) - Cv(D \mid A_i) \tag{4.7}$$

Using GPA data (Table 4.1), the detailed computations of *CvGain* are given below along with the conditional tables for $Cv(D \mid A)$. From Table 4.2,

$$Cv(D \mid A_1 = 2) = 0/3 * 100 = 0$$

From Table 4.3,

Table 4.4: Conditional table with $A_1 = 4$

Student	A_1	D
S3	4	1

$$Cv(D \mid A_1 = 3) = 0.4714/2.33 * 100 = 20.20$$

From Table 4.4,

$$Cv(D \mid A_1 = 4) = 0/1 * 100 = 0$$

Assuming that $P(A_i = a_j)$ is the probability that attribute A_i takes the value a_j,

$$\begin{aligned} Cv(D \mid A_1) = {} & P(A_1 = 2) * Cv(D \mid A_1 = 2) \\ & + P(A_1 = 3) * Cv(D \mid A_1 = 3) + P(A_1 = 4) * Cv(D \mid A_1 = 4) \end{aligned} \quad (4.8)$$

Hence,

$$Cv(D \mid A_1) = 1/5 * 0 + 3/5 * 20.2031 + 1/5 * 0 = 12.1219$$

$$CvGain(A1) = 34.0151 - 12.1219 = 21.8932$$

With similar calculations for attribute A_2,

$$Cv(D \mid A_2 = 1) = 0/3 * 100 = 0$$

$$Cv(D \mid A_2 = 2) = 0/2 * 100 = 0$$

$$Cv(D \mid A2 = 3) = 0.5/1.5 * 100 = 33.3333$$

$$Cv(D \mid A2) = 2/5 * 0 + 1/5 * 0 + 2/5 * 33.33 = 13.33$$

$$CvGain(A2) = 34.0151 - 13.3333 = 20.6818$$

As $CvGain(A_1)$ is large when compared with $CvGain(A_2)$, A_1 is selected as the splitting attribute.

4.4 Preprocessing

Preprocessing transforms the data such that it is suitable for building the data mining tools leading to effective knowledge discovery. A decision table may contain numerical and categorical attributes. A numerical attribute may be continuous by nature or may have high cardinality. To overcome the consequences of this type of numeric attributes, a discretization method is adopted.

Table 4.5: Illustration of the discretization process

A	Z	Z/k	floor(Z/k)	floor(Z/k)+T/k
4	-0.2221	-0.4442	-1	7
9	1.6285	3.257	3	11
3	-0.5922	-1.1844	-2	6
5	0.148	0.296	0	8
2	-0.9623	-1.9246	-2	6

A decision attribute may be numerical or categorical. As Cv can be computed for numerical data, a transformation is proposed for categorical decision attribute. The preprocessing techniques are detailed in the following.

conditional attributes

A numeric conditional attribute will be preprocessed (discretized) using the procedure outlined here.

1. *Normalization* The attribute values are transformed such that the attributes mean is 0 and standard deviation is 1. If x is the value of the attribute A with mean $\mu(A)$ and standard deviation $\sigma(A)$, then Z, the z-score normalized value of x is obtained as,
$$Z = \frac{x - \mu(A)}{\sigma(A)} \tag{4.9}$$

2. *Quantization* The attribute value is quantized with quantization parameter k. The attribute values must be translated to make them positive. The translation parameter T is used, the translation by T/k is performed on the nearest least integer value of the quantized value. Here the k value is chosen such that T/k is an integer. The resulting discretized value y is obtained as,
$$y = \left\lfloor \frac{Z}{k} \right\rfloor + \frac{T}{k} \tag{4.10}$$

This process is illustrated in Table 4.5, for values of attribute A, the normalized values are given in column Z, the discretized value of A is given in the last column. Based on the experiments, the values of the parameters are fixed as k=0.5 and T=4. The values 4,9,3,5 and 2, with mean 5 and standard deviation 3 are discritized as 7,11,6,8 and 6.

Decision attribute

The computation of *Cv* and *CvGain* require the decision attribute to be numerical with non zero mean. Hence any numerical coding with non zero mean is sufficient. The simplest such coding is, assigning a distinct positive integer for each value of the decision attribute. It is observed that the best coding for decision attribute is obtained by assigning the positive integers starting from 1 in non increasing order of the frequency of the decision attribute values. Hence the class with heighest frequency is assigned the code 1, while the class with second largest frequency is assigned the code 2 and so on. This preprocessing is required only for categorical decision attribute.

4.5 CvDT induction algorithm

The decision tree induction algorithm for constructing CvDT is shown in Algorithm 4. The input to this algorithm consists of the decision table DT with *attribute_list* and decision attribute *D*.

The algorithm works by recursively selecting the best attribute to split the data and expanding the leaf nodes of the tree until the stopping criterion is met. As all the decision trees available in the literature follow the fundamental algorithm proposed by Hunt [59] for tree construction, CvDT also follows the same. It works in similar manner to the standard decision tree induction algorithm (e.g. [53]), with the usage of *CvGain* for selection of attribute to split a node.

Algorithm 4: CvDT induction algorithm

```
CvDT (DT, attribute_list )
Input  : Decision table DT whose conditional attributes are attribute_list
         and decision attribute is D
Output: Decision tree
begin
    create a node N;
    if Cv(D) = 0 then
        return N as a leaf node labeled with class C, the class of all tuples;
    end
    if attribute _list is empty then
        return N as a leaf node labeled with the majority_class in D;
    end
    splitting_attribute ← arg max(CvGain(attribute_list));
    if CvGain(splitting_attribute) < 0 then
        return N as a leaf node labeled with the majority_class in D;
    end
    attribute_list ← attribute_list − splitting_attribute;
    // partition the tuples and grow sub trees
    foreach value j of splitting _attribute do
        DTj ← { tuples in DT with splitting_attribute = j} ;
        if DTj = φ then
            create a leaf node labeled with majority class in DT and attach
            it to node N;
        else
            attach the node returned by CvDT (DTj, attribute_list) to node N;
        end
    end
    return N;
end
```

Table 4.6: Decision Table for the Concept Play Tennis

Outlook	Temperature	Humidity	Windy	Class
Overcast	hot	high	false	N
Overcast	mild	high	true	N
Overcast	hot	normal	false	N
Overcast	cool	normal	true	N
Rain	mild	high	false	N
Rain	mild	high	true	P
Rain	cool	normal	false	N
Rain	mild	normal	false	N
Rain	cool	normal	true	P
Sunny	hot	high	false	P
Sunny	mild	high	false	P
Sunny	hot	high	true	P
Sunny	cool	normal	false	N
Sunny	mild	normal	true	N

4.6 Illustration

The popular Weather data set for the concept "Play Tennis" [90] is considered for illustration purpose (Table 4.6).

To compute *Cv*, the mean value need to be non zero. Hence the data need to be preprocessed in such a way that avoids mean to be zero. The proposed preprocessing procedure (section 4.4) is employed to preprocess the data of Table 4.6 Table 4.7 shown below is obtained after preprocessing Table 4.6 data to enable *Cv* computations.

CvGain values of each attribute are computed. These values are given in the Table 4.8,from the table it is observed that the attribute *Outlook* has maximum *CvGain*; hence the attribute *Outlook* is selected as splitting attribute at root node. Hence the data will be split into three sub tables as the attribute *Outlook* has three distinct values. For *Outlook* = *Overcast*, *Rain* and *Sunny*, the decision tables are shown in Tables 4.9, 4.10 and 4.11 respectively.

The corresponding building component of the decision tree is as shown in Figure 4.1.

With similar computations on Tables 4.9, 4.10 and 4.11 the decision tree is obtained as shown in Figure 4.2 with preprocessed codes replaced with their original values.

Table 4.7: Preprocessed decision table

Outlook	Temperature	Humidity	Windy	Class
Overcast	hot	high	false	1
Overcast	mild	high	true	1
Overcast	hot	normal	false	1
Overcast	cool	normal	true	1
Rain	mild	high	false	1
Rain	mild	high	true	2
Rain	cool	normal	false	1
Rain	mild	normal	false	1
Rain	cool	normal	true	2
Sunny	hot	high	false	2
Sunny	mild	high	false	2
Sunny	hot	high	true	2
Sunny	cool	normal	false	1
Sunny	mild	normal	true	1

Table 4.8: Cvgain Values

Attribute	*CvGain*
Outlook	5.73
Temperature	0.45
Humidity	2.42
Windy	0.74

Table 4.9: Decision Table For Outlook = Overcast

Temperature	Humidity	windy	class
hot	high	false	1
mild	high	true	1
hot	normal	false	1
cool	normal	true	1

Table 4.10: Decision Table For Outlook = Rain

Temperature	Humidity	windy	class
mild	high	false	1
mild	high	true	2
cool	normal	false	1
mild	normal	false	1
cool	normal	true	2

Table 4.11: Decision Table For Outlook = Sunny

Temperature	Humidity	windy	class
hot	high	false	2
mild	high	false	2
hot	high	true	2
cool	normal	false	1
mild	normal	true	1

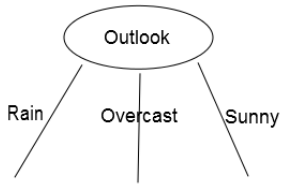

Figure 4.1: Decision tree with Outlook as splitting criteria at root node

4.7 Experiments and Results

The benchmark datasets from UCI [129] and CMU [29] are considered to demonstrate and validate the methodology and performance aspects of CvDT. The following Section 4.7.1 is devoted for decision systems where the information is available in decision tables. This validates CvDT as a decision tree. The next Section 4.7.2 is devoted for labeled text data sets. This illustrates CvDT suitability for text data.

4.7.1 Numeric data

To examine the effectiveness of our splitting criteria on decision tree construction, ten datasets are collected from UCI machine learning repository [129], shown in Table 4.12.

Decision trees are built with three different splitting criteria: Information Gain of ID3 [107], Gainfix of FID3 [38] and *CvGain* proposed in this chapter. The data sets with continuous values are discritized. The data sets are randomly permuted and tenfold cross validation is administered. Each time the same partitions of the data sets are used for building and testing the three decision trees compared. The philosophy of constructing decision tree algorithm is the same for all the three trees, only with difference in the selection criteria. Information Gain, GainFix and *CvGain* are used as the attribute selection criteria for ID3, FID3 and CvDT respectively. The classification performance and the times taken for training the decision trees are computed, as well as for testing them. Paired t-testis performed to verify the statistical significance of the results (a standard significance level of

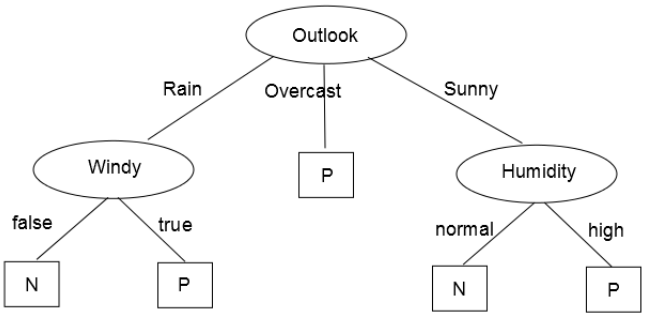

Figure 4.2: Final Decision tree

Table 4.12: Characteristics Of Data Sets

S.No	Data set	Number of Objects	Number of Attributes
1	Iris	150	4
2	Wine	178	13
3	WBC	699	10
4	Blood Transfusion	748	4
5	Abalone	4177	8
6	Ecoli	336	7
7	Yeast	1484	8
8	Page-blocks	5473	10
9	Wine red	1599	11
10	Pima-Indians Diabetes	768	8

0.05 is used). The characteristics of the datasets are shown in the table 4.12. The datasets collected contain 150 tuples as the least and 5473 tuples as the highest. The least number of attributes taken is 4 while the highest is 13. The results are shown in Table 4.13.

The advantage of *CvGain* is revealed in the times taken for decision tree generation. The generation times of CvDT are statistically significantly low when compared to the other two methods. In the case of the larger datasets considered in this experiment like Abalone and Page-blocks, the reduction of time is more clearly visible. With Abalone, 338 and 1672 milliseconds of time is saved when compared with ID3 and FID3 respectively. Similarly with Page-blocks they are 118 and 2726 milliseconds. Hence it is expected that *CvGain* is suitable for applications which require the decision trees to be built in real time. The Classification Performances are more or less equal for all the three trees. The observations based on the experiment are as follows:

1. The gain values of the attributes are different with *CvGain*, Information Gain and GainFix. But the attribute with maximum gain value is the same with all the three methods for some of data sets. It is different for some of

Table 4.13: Performance Comparison

Data set name	Classification Performance			Generation Time in ms			Testing Time in ms		
	CvDT	ID3	FID3	CvDT	ID3	FID3	CvDT	ID3	FID3
Iris	97.33	97.33	97.33	20.25	23.3	43.85	0.06	0.05	0.05
Wine	95.56	97.22	96.67	50.71	76.91	226.08	0.07	0.06	0.06
WBC	99.43	99.86	99.71	68.39	92.77	280.32	0.24	0.22	0.23
Blood Transfusion	81.81	81.67	81.81	131.06	153.24	264.64	0.34	0.33	0.33
Abalone	85.8	85.76	85.76	1586.44	1924.62	3258.47	18.01	18.26	21.1
Ecoli	95.88	95	95.29	123.71	168.67	319.79	0.13	0.13	0.13
Yeast	92.16	92.3	92.5	792.69	1002.69	1975.08	5.41	5.31	5.27
Page-blocks	97.15	97.28	97.44	785.26	903.79	3511.66	18.95	18.8	18.3
Wine red	95.63	95.19	95.31	756.97	1071.02	2741.15	5.39	5.51	5.07
Pima-Indians Diabetes	95.97	96.23	96.36	265.04	355.26	840.57	2.34	2.33	2.33

the data sets also.

2. It is possible that more than one attribute can have the maximum Gain value, and one of them is selected arbitrarily.

3. When the decision trees are verified, the decision trees built are the same for all the three approaches for the data sets Tom Mitchell, Iris and Blood Transfusion. The trees are different for the remaining eight data sets. For some of the data sets even though the decision trees are different, it is observed that some sub trees are the same. And few Attributes have interchangeable behavior in terms of selection for splitting because of the arbitrary tie breaking.

The algorithm of the FID3 paper is also implemented in Matlab environment, to make the readings comparable. (Thus the Classification Performances reported in the Table 4.13 are not the same as reported in [38]). The time taken for testing the CvDT is same as ID3 and FID3. But the time taken for generating decision tree using CvDT is significantly less when compared to ID3 and FID3. The paired t-test on generation times reveals statistical significance, indicating that CvDT construction time is significantly lower than ID3 and FID3. In fact CvDT is outperforming the other two methods in terms of generation time when the data sets are large in size. The following Figure 4.3 with the data sets along the x-axes reveals this.

4.7.2 Text Data

The "4 Universities Data Set" described in Chapter 3 is used to illustrate CvDT on text data. The vector space model as described in chapter 3 is used to form the conditional attributes, the class attribute has the values 'student', 'faculty', 'course' and 'project'. The conditional attribute values are discretized such that

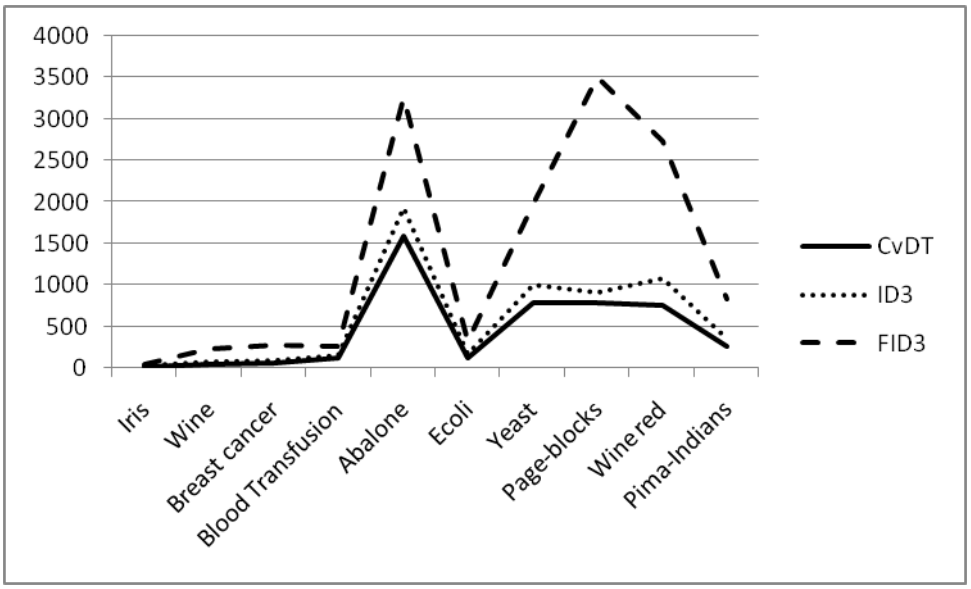

Figure 4.3: Comparison of Times taken for generating the three decision trees

each attribute mean is 0 and standard deviation is 1. The distribution of the class values are 'student' with frequency 1641, 'faculty' with frequency 1124, 'course' with frequency 930 and 'project' with frequency 504. Following the preprocessing proposed in this chapter, the decision attribute is assigned numerical coding to enable the *Cv* computation. The class values are given the numerical coding as 1 for 'student', 2 for 'faculty', 3 for 'course' and 4 for 'project'.

The term document matrix of this dataset has 7568 dimensions initially. Hence, dimensionality reduction is applied by following the method available in Chapter 3. The dimensionality is reduced to 948 by using linear regression for forming IA and Quick reduct [27] for finding a reduct. The accuracy of CvDT on this dataset is 0.94 on the reduced augmented decision table (as discussed in the section 3.3.2). It can be observed that the performance of CvDT is much better when compared to the best classifier using Support Vector Machine (c.f Table 3.11).

4.8 Salient features of CvDT

The salient features of *Cv* and hence CvDT are described in the following.

4.8.1 Low Computational Complexity

The computational complexity of *Cv* is low as it uses simple arithmetic operations for its computation. While entropy computation requires logarithmic function, *Cv* uses only arithmetic operations. It is even possible to avoid the usage of square root

function by considering variance instead of standard deviation in Cv computations for further reduction in computational complexity. This feature is clearly visible from the Figure 4.3, it is observed that the time taken for constructing CvDT is significantly lower.

4.8.2 Continuous Decision Attributes

CvDT can be used to build the decision tree even when the decision attribute is continuous. The decision trees which can predict the decision attribute value for continuous class attributes are normally called as regression trees. Some of the trees which are available in the literature that can predict the class value are [108], [109] and [14]. A survey on regression trees is available in [133].

CvDT can generate interval descriptions at the leaf nodes for such data (when decision attribute is nominal, point descriptions are generated). The algorithm for constructing CvDT using continuous decision attribute is given in Algorithm 5. This algorithm takes the decision table and α, a threshold on Cv, as input. When Cv is below α, a leaf node is created with the interval estimates (e.g. number of values, minimum value, maximum value, mean and standard deviation) of the data forming the leaf node. The remaining steps of the Algorithm 5 are similar to steps followed in Algorithm 4.

The Algorithm 5 performance is evaluated on CPU dataset [129]. The CPU dataset has 209 data objects and 9 attributes. Its decision attribute is "estimated relative performance" (ERP), which is a continuous attribute. CvDT is compared against CART [14] using the root mean squared error measure (RMSE). Average RMSE over ten folds (ten fold cross validation) for CvDT is 39.52 and 47.89 for CART. The CvDT RMSE value is obtained using the parameter α as 33%, based on a rule of thumb from [119]. Infact α can be as low as 15%, the variation of

RMSE with α is shown in Figure 4.4

Algorithm 5: CvDT induction algorithm for continuous decision attribute

```
CvDT (DT, attribute_list ) Input : Decision table DT whose conditional
        attributes are attribute_list and decision attribute is D, α which is
        a threshold on Cv
Output: Decision tree
begin
    create a node N;
    if Cv(D) < α then
        return N as a leaf node labeled with the mean and interval
        estimates of D;
    end
    if attribute _list is empty then
        return N as a leaf node labeled with the mean and interval
        estimates of D;
    end
    splitting_attribute ← arg max(CvGain(attribute_list));
    if CvGain(splitting_attribute) < 0 then
        return N as a leaf node labeled with the mean and interval
        estimates of D;
    end
    attribute_list ← attribute_list − splitting_attribute;
    // partition the tuples and grow sub trees
    foreach value j of splitting _attribute do
        DTj ← { tuples in DT with splitting_attribute = j} ;
        if DTj = φ then
            create a leaf node labeled with most probable value in DT and
            attach it to node N;
        else
            attach the node returned by CvDT (DTj, attribute_list) to node N;
        end
    end
    return N;
end
```

Using Monte Carlo simulation for hundred times, with ten fold cross validation in each iteration of the simulation, the distributions of RMSE values of CvDT and CART are compared. The distribution of RMSE of CvDT with $\alpha = 33\%$ is having

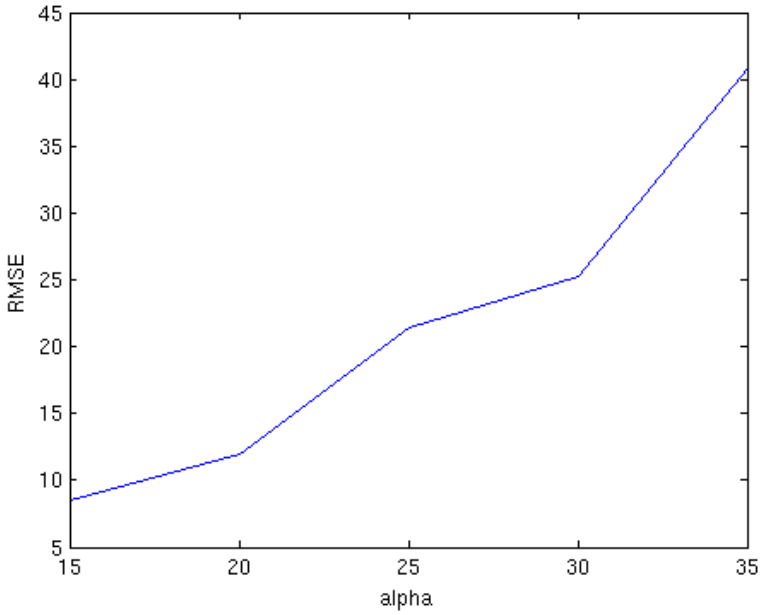

Figure 4.4: Variation of RMSE with *alpha*

mean = 38.57 with standard deviation = 1.1732. The distribution of RMSE of CART is having mean = 45.69 with standard deviation = 4.6299. Figure 4.5 shows the comparison of these two trees performance in terms of RMSE. It can be observed that CvDT has low RMSE values with less deviation.

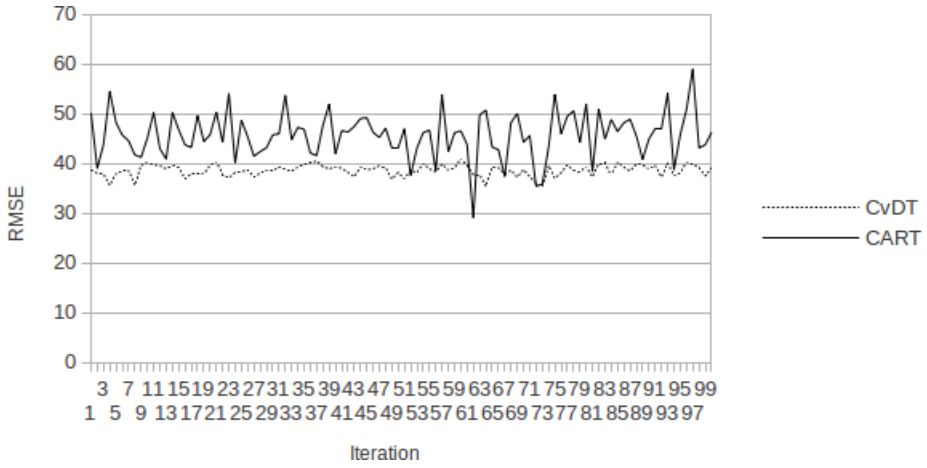

Figure 4.5: RMSE of CvDT and CART

4.8.3 Dependency on Measurement

Entropy depends on frequency alone, ignoring the measurement. Whereas *Cv* depends on measurement also. SD and variance work with absolute values whereas *Cv* is a relative measure and free from units. Entropy computation requires the

decision attribute to be nominal. *Cv* computation is possible only when measurements are available. The decision attribute has to be real and positive for *Cv*. A preprocessing technique as given in section 4.4 addresses nominal, categorical and continuous decision attributes. This preprocessing approach is not an overhead as it depends mainly on the meta data available with the decision system. For a 2-class problem, the comparison of Entropy, Gini Index and *Cv* are shown in Figure 4.6. The skewness in *Cv* is attributed to its dependency on measurement.

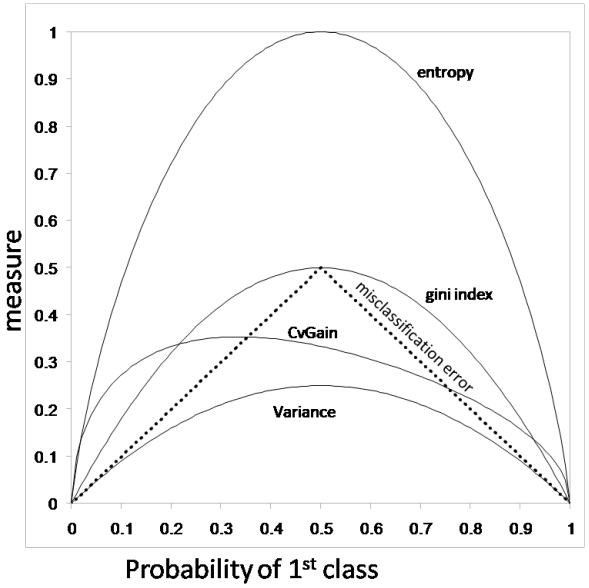

Figure 4.6: Behavior of attribute selection measures

4.8.4 Distributed Computation

Distributed decision trees are needed when the data is large or when the data is distributed. Distributed decision trees available in the literature are constructed by using horizontal or vertical fragmentation of the data. CvDT can be constructed in distributed environments with these fragmentation approaches due to the algebraic characteristics of *Cv*. Decision tree induction from distributed data sources as proposed in [48] and [19] serve as examples. CvDT can be constructed using either of the fragmentation approaches. The procedures for these are outlined in the following.

Distributed CvDT induction with horizontal fragmentation of data

With horizontal fragmentation, subsets of data tuples are stored at different sites. Assume that the data set D is available in (divided into) k fragments, D_1, D_2,..., D_k. Each site will have data for all attributes. The triplet

1. number of tuples
2. sum of decision attribute values of the tuples
3. sum of squares of decision attribute values of the tuples

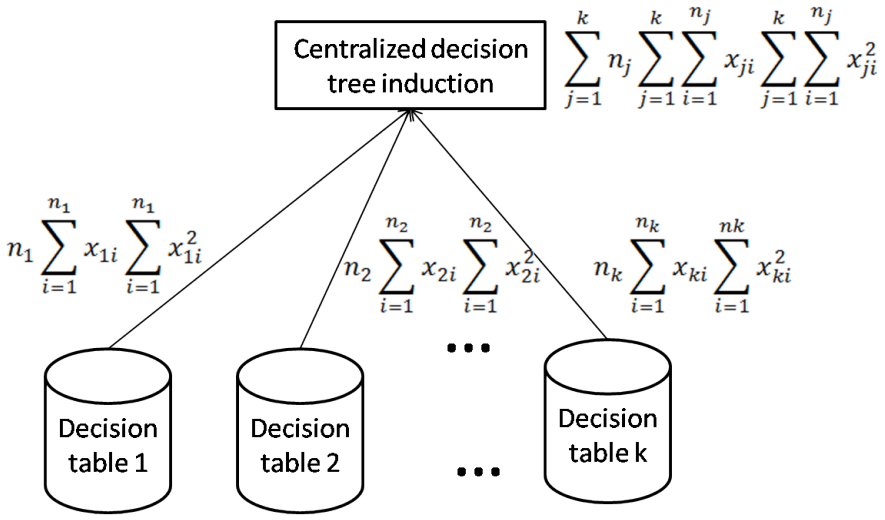

Figure 4.7: Distributed CvDT induction with horizontal fragmentation of data

is computed at each site, for site (fragment) j as $< n_j, \sum_{i=1}^{n_j} x_{ji}, \sum_{i=1}^{n_j} {x_{ji}}^2 >$. The coefficient of variation of the entire data is computed at the centralized computation, using these triplets from all the sites,

$$\text{number of tuples, } n = \sum_{j=1}^{k} n_j \tag{4.11}$$

$$\text{sum of decision attribute values, } sum = \sum_{j=1}^{k} \sum_{i=1}^{n_j} x_{ji} \tag{4.12}$$

$$\text{sum of sqaures of decision attribute values, } sumsq = \sum_{j=1}^{k} \sum_{i=1}^{n_j} {x_{ji}}^2 \tag{4.13}$$

When *CvGain* of an attribute has to be computed, the triplet is taken from each site for each value in the domain of the attribute. This facilitates the *CvGain* computation for all the attributes, hence the CvDT induction at the central site.

Distributed CvDT induction with Vertical fragmentation of data

With vertical fragmentation, each site will have all objects with a sub set of attributes. Each site will have a subset of conditional attributes and the decision attribute. The coefficient of variation of the entire data is computed using the triplet of any site (as each site has all tuples for the sub set of the attributes). Each site will compute the Cv and CvGain and shares them with the central node. The best attribute and associated object partitions wil be available in the shared memory for further computations.

4.9 Conclusions

The criterion for splitting a node in a decision tree decides the efficiency of a decision tree. So far, Information Gain, Gain ratio, Gini index, Chi square statistic and Kappa index are used as the splitting criteria. *CvGain* is proposed and demonstrated as another splitting criteria in this paper. It is based on Coefficient of Variation (Cv), which is a measure of consistency of a distribution. It has been observed that decision tree based on *CvGain* has the same performance as ID3 and FID3, but at less computational cost. This is proved by the experiments on benchmark datasets and the hypothesis test performed with paired t-test. Hence it is suitable for agent based applications, where a decision tree has to be built in real time.

By imposing the stopping criteria on Cv at decision node, post-pruning may be avoided. Adaptability and the limitations of CvDT in various paradigms is in progress.

CHAPTER 5

Greedy Incremental Clustering Approach (GICA)

A clustering methodology which is greedy and follows incremental approach akin to snowball in growing the clusters is described in this chapter. This method of forming clusters and its suitability to various kinds of data is demonstrated with validations. In particular, its suitability for search results clustering is analyzed empirically.

5.1 Introduction

The Search Results Clustering systems mainly differ in arriving at the cluster description and the cluster content [22]. As the initial clusters displayed have to be the most prominent sub topics of the query, a greedy approach is proposed for this purpose. This chapter describes the procedure followed to identify the subtopics of the query which forms the clusters' description and their content. Further, the proposed clustering algorithm's ability to handle numeric data is demonstrated. The clustering algorithm can be feature centric or object centric, the feature centric approach is suitable for SRC while the object centric approach is suitable for numeric data. Two clustering approaches are developed due to the special requirement associated with SRC, the cluster labels need to be meaningful descriptions of the clusters. In fact, the feature centric approach is applicable for text data, especially for document clustering.

These two approaches for clustering are described in this chapter. The findings of [47] has shown that frequency based feature ranking is not suitable for SRC. Hence Expectation Maximization based adaptive feature pruning method is proposed. The features not pruned by this method are called relevant. Knowledge preserving capabilities of these relevant features and Snowball clustering method based on these relevant features is demonstrated in this chapter.

5.2 Object Centric GICA

The classical clustering algorithms like k-means [81] and k-medoids are object centric [53]. This section develops an object centric greedy incremental algorithm and demonstrates the same.

The vector median of the data objects to cluster is the candidate to initiate the clustering process. The objects which are in the neighborhood (object distance is less than the adaptive threshold) of the vector median object are added to the cluster. The objects which are in the neighborhood of these objects are added in the growing phase of the cluster in incremental manner. When the cluster can not grow further, next cluster is initiated by picking up the vector median of the remaining objects and this process is continued till all objects are exhausted. This procedure identifies many small pure groups of objects. According to [86], one manually assigned class gets divided into small pure groups by the clustering algorithms. Hence, the initial clusters formed by the proposed method are combined/merged in agglomerative manner. These clusters are merged by following Single linkage method [53]. The termination criteria for the agglomeration process is arrived by EM algorithm [36] and makes this agglomeration process totally machine learning.

5.2.1 Preprocessing

The clustering algorithm needs neighborhood information. Neighborhoodness depends on distance. Numerical attributes can have varying ranges of values, hence the threshold for neighborhoodness keeps varying. Normalization is useful to prevent the attributes with large range of values dominating the attributes with smaller ranges. Normalization is particularly useful when distance measurements are involved [53]. Hence the data is normalized using *min-max normalization* to the range of 0 to 1. Normalization to this range enables the neighborhood measure to be uniform for all the datasets.

5.2.2 Object Centric GICA Algorithm

The first step of the Algorithm 6 is discretization. The conditional attributes are discretized using the procedure described in Section 4.4 of chapter 4. VM is the vector median of the data objects. A cluster is initiated with the VM object. The threshold θ_1 is arrived by using the mean (or standard deviation) of the VM object's distance to all the remaining objects. All objects, whose distance from

VM is below θ_1 are added to the cluster. The cluster is grown in incremental manner by addition of objects in the neighborhood of objects of the cluster. This neighborliness is defined by using threshold θ_2 which is obtained based on θ_1. When the cluster can not grow further, the next cluster (j is incremented) is initiated with the vector median of remaining objects. The process is continued until all objects are exhausted, i.e $|objects| = 0$.

The clustering procedure forms many clusters with high purity. So these clusters are merged based on their similarity. Expectation Maximization [36] algorithm is used to arrive at threshold θ_3 on distances between each pair of clusters. The clusters whose distance is below θ_3 are considered similar and they are merged. The merging process is based on Single linkage agglomeration. As described, all the thresholds θ_1, θ_2 and θ_3 are obtained based on the data characteristics without user specification.

Algorithm 6: Object Centric GICA

```
Input  : D is an array of data objects to cluster
Output: clusters where each cluster is a set of objects
begin
    Discretize (D);
    j ← 1;
    repeat
        VM ← VectorMedian (objects);
        objects ← objects − VM;
        cluster_j ← {VM} ;
        for i ← 1 to |objects| do
            if distance (object_i, VM) < θ_1 then
                cluster_j ← cluster_j ∪ object_i;
                objects ← objects − object_i;
            end
        end
        // growing phase of the cluster
        repeat
            clusterSize ← |clusters_j|;
            for i ← 1 to |objects| do
                for k ← 1 to |cluster_j| do
                    if distance (object_i, object_k ∈ cluster_j) < θ_2 then
                        cluster_j ← cluster_j ∪ object_i;
                        objects ← objects − object_i;
                    end
                end
            end
        until |cluster_j| = clusterSize;
        j ← j+1;
    until |objects| = 0;
    // merging phase of the cluster
    Dmin ← minDistance(clusters);
    θ_3 ← EM (2,Dmin);
    clusters ← SingleLinkageAgglomerative (clusters,θ_3);
end
```

5.2.3 Illustration

A sample from Iris dataset [129] is selected for illustrating the Object centric GICA algorithm. The dataset is shown in the Table 5.1. The Class attribute information is used only for evaluating the clusters. Initially object 14 is identified as vector

Table 5.1: Iris data for demonstrating Object centric GICA

S.No	Sepal length	Sepal width	Petal length	Petal width	Class
1	6.1	3	4.6	1.4	2
2	7.7	2.6	6.9	2.3	3
3	6.2	2.8	4.8	1.8	3
4	5	3.5	1.6	0.6	1
5	5.1	3.8	1.5	0.3	1
6	6.6	3	4.4	1.4	2
7	4.9	3.1	1.5	0.1	1
8	6	2.2	5	1.5	3
9	6.9	3.1	4.9	1.5	2
10	6.5	2.8	4.6	1.5	2
11	4.6	3.2	1.4	0.2	1
12	5	2.3	3.3	1	2
13	5.8	2.6	4	1.2	2
14	6.4	2.9	4.3	1.3	2
15	6.1	2.9	4.7	1.4	2
16	7.6	3	6.6	2.1	3
17	6.8	2.8	4.8	1.4	2
18	6.3	2.8	5.1	1.5	3
19	5.8	2.7	5.1	1.9	3
20	5.4	3.7	1.5	0.1	1

median, and the threshold values obtained are $\theta_1 = 0.1535$ and θ_2=0.0076. The objects 1,6,10 and 15 are added during the growing phase of the cluster1. Cluster2 is initiated with object 13 (whose thresholds are 0.1986 and 0.0099). The object 17 is added to cluster2 during its growing phase. Proceeding similarly, 12 clusters are formed. These initial clusters have Purity 1.0, Entropy 0.0, Precision 1.0 and Recall 0.25.

These clusters are merged with Single linkage agglomeration. The pair wise distances of the clusters are fed to EM algorithm to find the cutoff for guiding the stopping criteria of merging. The cutoff identified on the cluster distances is 1.014. After merging the clusters, the crosstable based on the Class attribute is tabulated in Table 5.2 From the cross table 5.2 the validation measure are
Entropy = 0.00 Purity = 1.0 precision = 1.00 recall = 1.00.

Table 5.2: Cross table for Iris demo data clusters

	Class1	Class2	Class3
Cluster1	0	9	0
Cluster2	0	0	6
Cluster3	5	0	0

5.2.4 Experiments and Results

The datasets for validating the clustering algorithm are collected from UCI machine learning repository [129]. The datasets collected are having class information which can serve as ground truth for validating the clustering algorithm's performance. The results are reported in Table 5.3.

Table 5.3: Object centric GICA performance metrics

Data set	Entropy	Purity	Precision	Recall	F-score
Iris	0.00	1.00	1.00	1.00	1.00
Wine	0.00	1.00	1.00	0.75	0.86
WDBC	0.04	0.99	0.99	0.40	0.57
Blood Transfusion	0.00	1.00	1.00	0.40	0.57
Abalone	0.35	0.88	0.93	0.28	0.43
Ecoli	0.70	0.81	0.83	1.00	0.91
Yeast	1.54	0.60	0.65	1.00	0.79
Page-Blocks	0.29	0.93	0.92	0.65	0.76
Wine Quality Red	0.00	0.99	1.00	0.36	0.53
Pima-Indians Diabetes	0.00	1.00	1.00	0.18	0.31

To compare the proposed method with K-Means clustering method [81, 83], K-Means is run on the same datasets and the results are provided in Table 5.4. Two fundamental differences exist between these two algorithms. K-Means is supplied with the number of clusters existing in the data and its clustering performance varies with the initial centroids used. Such dependencies do not exist in the proposed clustering algorithm, it follows machine learning approach and does not require any knowledge about the number of clusters. The performance of the proposed method does not vary for each execution as it does not depend on initialization in contrast to K-means.

5.3 Feature Centric GICA

This clustering algorithm develops the concept and content for each cluster in greedy incremental fashion. The primary motivation for developing this clustering

Table 5.4: K-Means clustering performance metrics

Data set	Entropy	Purity	Precision	Recall	F-score
Iris	0.00	0.98	1.00	1.00	1.00
Wine	0.65	0.71	0.86	0.67	0.75
WDBC	0.11	0.98	0.98	0.99	0.98
Blood Transfusion	0.00	0.99	1.00	1.00	1.00
Abalone	0.41	0.83	0.84	0.52	0.64
Ecoli	0.30	0.89	0.93	0.50	0.65
Yeast	0.22	0.92	0.89	0.60	0.72
Page-Blocks	0.06	0.98	0.90	0.60	0.72
Wine Quality Red	0.18	0.95	0.94	0.50	0.65
Pima-Indians Diabetes	0.00	0.99	1.00	1.00	1.00

approach is to develop clusters with meaningful labels which suits the primary requirement of SRC. The same clustering approach is applicable for text data in general. Hence, the following description and the validation are given in the context of SRC.

Each search result consists of four fields, namely result ID (rank of the result), url of the result web page, title of the result web page and short description of the result web page (snippet). The title and snippet are used for the clustering process, these constitute search result document for the clustering process. The search results are preprocessed by stop-word removal, stemming and stray HTML tags removal. The feature selection process identifies prominent features (terms in the current context) and closed frequent itemset analysis identifies the dominating topics. The clusters are formed by greedy approach for the best (most prominent) closed frequent itemset to initiate a cluster. The itemset with the highest support count is the most prominent subtopic of the query. Each cluster has two components - concept of the cluster and the content of the cluster. Concept of the cluster is formed by the set of closed frequent itemsets of the cluster. Content of the cluster is formed by the search results of the cluster. The search results that contain the greedily picked up closed frequent itemset form the initial content of the cluster. The cluster is grown in incremental fashion by adding the closed frequent itemsets that are prominent in the cluster content, and adding the search results that contain the newly added itemsets. This growing phase of a cluster is stopped when the cluster can not grow further and the prominent itemset out of the remaining is picked up to initiate next cluster. This procedure to cluster stops, when either the closed frequent itemsets or search results are exhausted.

The label of the cluster is the itemset with highest support among the itemsets of the cluster. As the itemset contains stemmed terms, their corresponding unstemmed termsets are obtained from the search results.

5.3.1 Preprocessing

Some preprocessing tasks are performed prior to using the search results for clustering. These tasks are stop-word removal, stemming, handling of digits and punctuation symbols, cases of letters and HTML tags removal.

Stop-word removal is performed on each search result's title and snippet. The stop-word list of 571 words from SMART system [60] is adopted for stop-word removal. Martin Porter's suffix stripping algorithm[104] is used for stemming.

Each search result document is represented as a vector of tf-idf weights.

5.3.2 Datasets

AMBIENT

AMBIENT (AMBIgous ENTries) is a dataset designed for evaluating subtopic information retrieval. It consists of 44 topics, each with a set of subtopics and a list of 100 ranked documents. The topics were selected from the list of ambiguous Wikipedia entries; i.e., those with "disambiguation" in their title [127]. The 100 documents associated with each topic were collected from a Web search engine as of January 2008, and they were subsequently annotated with subtopic relevance judgments. The AMBIENT dataset consists of four files where each row is terminated by Linefeed (ASCII 10) and fields are separated by Tab (ASCII 9). The four files are described below.

topics.txt

It contains topic ID and description. Sample content is shown in table 5.5

Table 5.5: Topics sample of AMBIENT

ID	Description
1	Aida
2	B-52
3	Beagle

subTopics.txt

It contains subtopic ID (formed by topic ID and subtopic number) and description. Sample content is shown in Table 5.6

results.txt

The file is obtained directly by the search engine API and characters are

Table 5.6: Sub Topics sample of AMBIENT

ID	Description
1.1	Aida, female given name
1.2	Aida can be a Japanese name
1.3	A Persian name

coded using UTF8 (a variable-length character encoding for Unicode). e.g. the sequence 195 169 codifies the small letter e with acute accent. It contains result ID (formed by topic ID and search engine rank of result), url, title, and snippet. Sample content is shown in Table 5.7.

Table 5.7: Results sample of AMBIENT

ID	url	title	snippet
1.1	http://www.aida-international.org/	AIDA International	International Assoication for Development of Apnea dedicated for breath-hold diving or apnea which manages and oversees the recognition of records, organizes competitions, and sets standards for freedive education.
1.2	http://disney.go.com/theatre/aida	Disney on Broadway - 'Elton John and Tim Rice's Aida'- Official Homepage	The official site of 'Elton John and Tim Rice's Aida.' Buy tickets online, meet the cast, hear the Tony Award-winning music, get performance and tour information, ...
1.3	http://en.wikipedia.org/wiki/Aida	Aida - Wikipedia, the free encyclopedia	For other references see Aida (disambiguation) Aida is an opera in four ... Aida, the daughter of the Ethiopian King Amonasro, lives at Memphis as a slave. ...

STRel.txt

It contains subtopic ID (formed by topic ID and subtopic number) and result ID (formed by topic ID and search engine rank of result). Sample content is shown in table 5.8.

Table 5.8: Topic.Subtopic ID of the Result

subTopicID	resultID
1.4	1.3
1.4	1.4
1.4	1.10

MORESQUE

MORESQUE (MORE Sense-tagged QUEries) [94] is a dataset designed for evaluation of subtopic information retrieval. The dataset consists of 114 topics (i.e.,

queries), each with a set of subtopics and a list of 100 top-ranking documents. MORESQUE is designed as a complement for AMBIENT. This dataset also contains four files as in AMBIENT dataset. This dataset contains queries of 1 to 4 words while the AMBIENT dataset is composed mostly of single-word queries.

5.3.3 Feature Selection - Expectation Maximization

Frequency based feature ranking alone is ineffective to cluster the web search results [47]. Terms with discriminative capability are effective for SRC. Hence, Expectation Maximization [36] is used to prune the irrelevant terms to make this approach meet the SRC requirements. The size of the relevant terms set due to EM makes the frequent itemset analysis (the clustering algorithm uses frequent itemsets) manageable, though frequent itemset analysis is a tedious task.

Expectation Maximization is frequently used for data clustering in Machine learning, as it can handle latent variables in the data. Expectation Maximization method is developed for characterizing the population characteristics, which is a mixture of finite fundamental factors. Each factor will have its own uncertainty model characterized by associated probability distribution. The characteristics of the population will obey an uncertainty model characterized by mixed distribution, which is generalization of fundamental factors. This mixture model will have set of parameters, namely mixing proportions, and the parameters of fundamental factor distributions. If one assumes that the population is a mixture of k fundamental factors, and each factor obeys normal distribution, when the characteristic under study is 1-dimensional, then the mixture model will have parameter's mixing proportions $\pi_1, \pi_2, \ldots, \pi_k$ ($\pi_1 > 0$ and $\sum_{i=1}^{k} \pi_i = 1$) with mean π_i and standard deviation σ_i for i^{th} fundamental factor. Estimation process of these parameters is addressed by EM based on the sample [36].

The terms occurring in each preprocessed search result constitute the population for the analysis. However, some of the terms are relevant and some are not relevant for building the knowledge. Thus, the distribution of the TF-IDF values of the terms can be viewed as a mixture of relevant and irrelevant terms (i.e. as a mixture of two distributions). The simpler way of representing it is by using Gaussian distribution, $f(x) = \pi_1 f_1(x|\mu_1, \sigma_1) + \pi_2 f_2(x|\mu_2, \sigma_2)$, where f_1 is the probability distribution function of TF-IDF values of relevant terms with mean μ_1 and and standard deviation σ_1, f_2 is the probability distribution function of TF-IDF values of irrelevant terms with mean μ_2 and and standard deviation σ_2; π_1 and π_2 are mixing proportions. These parameter estimates must have the property that $\mu_1 > \mu_2$. Based on the parameters obtained for the mixture model, the threshold

on TF-IDF values is $\frac{\pi_1\mu_1+\pi_2\mu_2}{\pi_1+\pi_2}$, from [3]. Any term with TF-IDF value more than this threshold is classified as relevant term, otherwise irrelevant term.

Illustration of EM based Feature Selection

An illustration of EM based term pruning is provided by considering the first ten results of the query "Beagle" from the AMBIENT dataset.

The first ten results after tokenization (using white space as delimiter) resulted in 189 terms:

> *(EC), (software), -, –, ..., 2, 3, 5, :, A, ANSI/ISO, American, Apple, Architecture., BEAGLE, Beagle, Beagle,, Beagle-type, Beagle., Beagles, Breed, Britannica, British, C++, C++., Center,, Club, Computation, Consortium, Debian, Desktop, Dog, Download, Earth., Encyclopaedia, English, Evolutionary, Files., GNOME, God's, Google, Group., Guide, Head., Hound, Information, Information, Kennel, Linux, Main, Mars, Open, Owner's, PSSRI, Package, Page, Pillinger , Professor, Profile:, Search, Size, Spotlight, Standard., Storage, The, There's, University-based, University., Unix, W3, Wikipedia,, Windows, a, all, also, among, an, and, any, architectures., are, article, as, at, available, be, beagle, beagle., blog-post,, breed, built, but, by, can, centuries,, child,, coat,, code, coded, color, come, compliant, critters, cutest, data, dog, dog,, dogs, easy-care,, encyclopedia, enhance, entirely, existed, experience, exploration, fairly, for, framework, free, get, good, gotten, green, group,, has, have, head, heavy, hound, idea, in, indexing, industrial, is, its, led, long,, medium, member, modern, not, of, on, or, other, over, overwhelmed, pack, page, partners, people, personal, pet, point,, post, power, product, puppies, puppy, reference, researchers, search, short, should, similar, simplicity, single, sized, skull, sleek,, small, solidly, sporty, standard., system, the, their, this, to, together, tool, website,, where, which, will, with, you, your*

The preprocessing by stemming and stop-word removal resulted in 118 terms:

> *american, ansiiso, appl, architectur, articl, avail, beagl, beagletyp, blogpost, breed, britannica, british, built, center, centuri, child, club, coat, code, color, come, compliant, comput, consortium, critter, cutest, data, debian, desktop, dog, download, earth, easycar, ec,*

encyclopaedia, encyclopedia, english, enhanc, entir, evolutionari, exist, experi, explor, fairli, file, framework, free, gnome, god, good, googl, gotten, green, group, guid, head, heavi, hound, idea, index, industri, inform, kennel, led, linux, long, main, mar, medium, member, modern, open, over, overwhelm, owner, pack, packag, page, partner, peopl, person, pet, pilling, point, post, power, product, professor, profil, pssri, puppi, refer, research, search, short, similar, simplic, singl, size, skull, sleek, small, softwar, solidli, sporti, spotlight, standard, storag, system, togeth, tool, univers, universitybas, unix, w3, websit, wikipedia, window

Using vector space model, TF-IDF scores are computed for the preprocessed result documents. When EM algorithm is run, the threshold on TF-IDF values is obtained as 1.17, based on the method described. It is observed that 28 terms have TF-IDF values above this threshold. These terms are treated as relevant terms and they are given below:

american, architectur, britannica, british, club, code, debian, desktop, download, encyclopedia, explor, free, guid, inform, kennel, led, main, mar, open, owner, page, profil, puppi, search, softwar, w3, wikipedia

These relevant terms form the feature set. The average terms set size for all the 44 queries of AMBIENT dataset, resulted without term pruning as 865.18 (with standard deviation 74.06), and with term pruning as 274.22 (with standard deviation 72.22). The percentage of reduction in terms set size is 68.30.

The "degree of knowledge representation" (Kappa) developed in the Rough Set theory [101], which quantifies the knowledge representation in considered features is employed, treating the terms as features. It is observed that pre pruning produced mean Kappa as 1 (SD = 0), indicating that the pre-pruned terms set possess hundred percent knowledge about the ground truth. Whereas post pruning term set has mean Kappa 0.95 (SD=0.04) indicates that, the knowledge representation by post pruned term set will possess the knowledge about the ground truth from 0.8 to 1. The relevant terms set may lead at the most 20% loss in knowledge representation, 68.02% gain by reduction in term set size. This suggest that, the relevant term set obtained by EM based term set pruning can be employed in search result analysis in Web Mining.

EM based term pruning has two benefits: overcome the curse of dimensionality and less runtime memory.

5.3.4 Frequent Itemset generation

Frequent patterns are useful to identify the associations and correlations existing in the data. Frequent patterns help in data classification and clustering [53]. Frequent itemsets are used for document clustering in [46]. Document clustering using frequent termsets (each item in conventional itemset mining is a term) has the following benefits: dimensionality reduction, independence from the specification of number of clusters and meaningful cluster description [46]. Search results clustering works on text data (title and snippet of the search results) and these aspects are not considered in the existing SRC approaches.

The frequent termsets are identified by FPGrowth [7], by considering the relevant terms retained after the EM based term pruning.

Document clustering algorithm using frequent itemsets is proposed in [9]. It is reported that the clustering quality is comparable to the state of the art text clustering algorithms and the frequent term sets provided an understandable description of the clusters. Document clustering proposed in [46] is reported to have best clustering quality and scalability, while frequent itemsets can reduce dimensionality and provide specific topics of the documents. Term Document Clustering (TDC) proposed in [144] uses only closed frequent itemsets (a frequent itemset is closed if it has no superset with the same frequency). It has better clustering accuracy and scalability due to further dimensionality reduction. The quality of these algorithms is subjective to the minimum support value, and TDC can estimate the suitable minimum support value. Hierarchical Clustering using Closed Interesting Itemsets proposed in [84] uses only closed interesting itemsets. It uses Mutual Information, Chi Square, etc as the interesting measures. But this causes loss of information when closed itemsets are reduced. As SRC deals with small text snippets, this method will further induce loss of information; hence only closed frequent termsets are used.

However, frequent itemset based clustering methods are sensitive to minimum support threshold. A wise choice of minimum support results in a better clustering performance. Authors of [144] proposed a method to select minimum support. It uses FT tree which is based on FP tree of [54] and document coverage information.

Hence, FT tree has been employed for Search Result Clustering for arriving at the support threshold adaptively. FT tree is built as proposed in [144]. Document coverage distribution for different support values is constructed by pruning the FT tree. Minimum support threshold, min_sup is determined from the above distribution. The FT tree is prunedby using the min_sup. Frequent closed termsets are obtained by employing CLOSET which is developed by [102].

The associated FT Trees of pre and post term pruning with EM, have been constructed and the corresponding estimates of the size (obtained by multiplying the maximum depth and maximum width) are derived and reported in Figure 5.1. For the 44 queries data of AMBIENT, the Mean (Standard Deviation) of FT Tree sizes before and after term pruning are 2486.48 (SD = 286.12) and 962.82 (SD = 243.29) respectively. The percentage of reduction in FT tree sizes is 61.28.

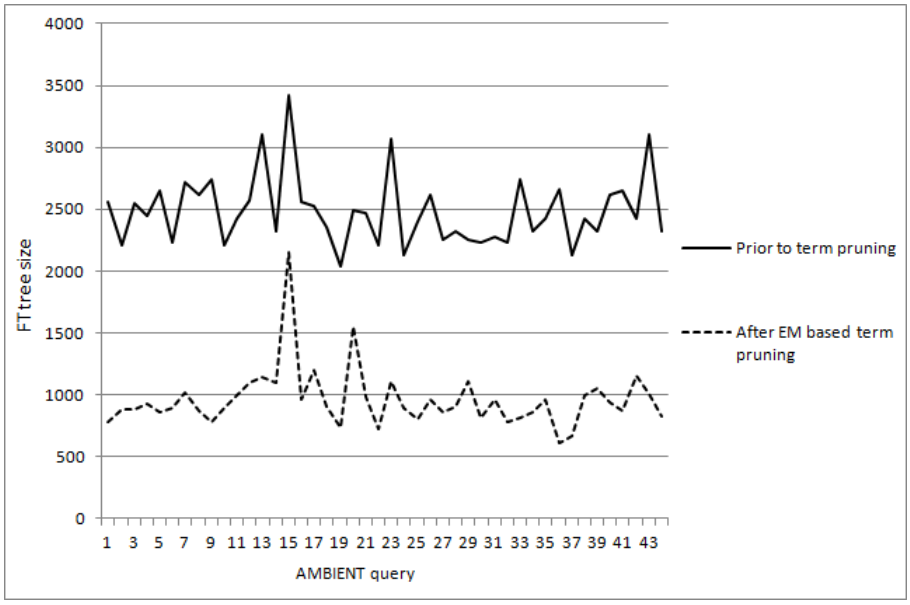

Figure 5.1: FT tree sizes before and after term pruning

5.3.5 Feature Centric GICA Algorithm

The Feature Centric GICA Algorithm is described in Algorithm 7. The first step of the algorithm is *FeatureSelection*, to select the features to form the concept for the cluster. The data objects which possess these features are added to the cluster content. These features and objects that form the cluster are removed from the original set of features and objects. To generate overlapping clusters, this step has to be bypassed. The cluster is grown in incremental fashion by repeatedly adding features and objects as long as the cluster is growing. The growing phase is guided by percentage of objects containing the features of the cluster. When a cluster can not grow further, next cluster is initiated by performing *FeatureSelection* and the process continues till the features are exhausted.

Algorithm 7: Feature Centric GICA Algorithm

Input : D is an array of Search Results to cluster, F is the feature set of D

Output: clusters which is a set of clusters, each cluster is a record of features and objects

```
begin
    j ← 1;
    repeat
        CF ← FeatureSelection (F);
        baseclusters_j.features ← CF;
        baseclusters_j.objects ← {objects of D conditioned on CF };
        F ← F − CF;
        Objects ← Objects − baseclusters_j.objects;
        repeat
            baseclusterSize ← |baseclusters_j.objects|;
            // growing phase, add features covered in the
               basecluster
            for f ← 1 to |F| do
                if percentage of baseclusters_j.objects containing F_f > θ_1 then
                    F ← F − F_f;
                    CF ← CF ∪ F_f;
                end
            end
            baseclusters_j.features ← CF;
            // growing phase, add objects through the added
               features
            for f ← 1 to |baseclusters_j.features| do
                newObjects ← {objects of D conditioned on
                baseclusters_j.features_f};
                Objects ← Objects − newObjects;
                baseclusters_j.objects ← baseclusters_j.objects ∪ newObjects;
            end
        until |baseclusters_j.objects| = baseclusterSize;
        j ← j+1;
    until |F| = 0;
end
clusters ← merge (baseclusters, j);
```

5.3.6 Illustration

The documents used for illustration are shown in Table 5.9. After the prepro-

Table 5.9: Document set for demonstration of FCGICA

Document	content
Document1	Apple mac
Document2	iphone from Apple
Document3	water molecule
Document4	Osmosis of water
Document5	iphine tunes
Document6	Apple mac price
Document7	Osmosis of water molecule
Document8	reverse Osmosis of water molecule

cessing step (stop-word removal and stemming are applied but EM based feature selection is not performed in this this illustration for simplicity). The FT tree of the documents is shown in Figure 5.2. The document coverage table from the FT tree is shown in Table 5.10. It describes the coverage information for each support value. When there is a knee in the document coverage, the pruning of FT tree is stopped and the support threshold is picked up as the value prior to the knee in coverage.

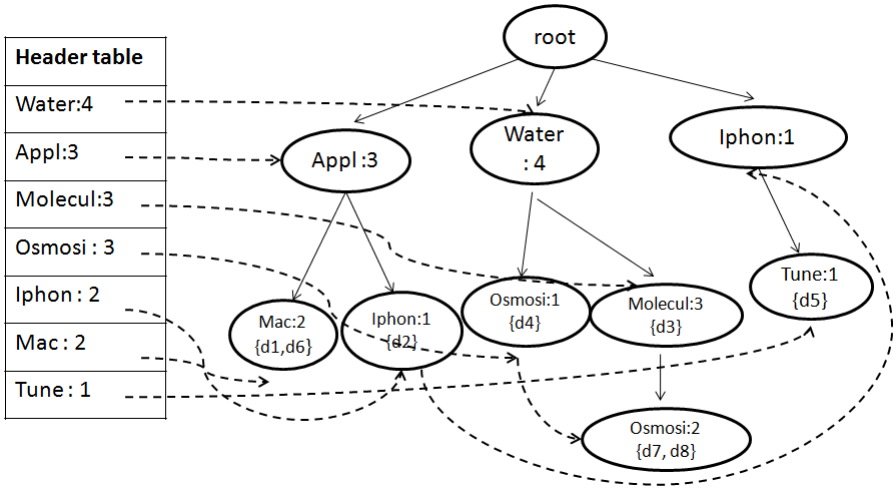

Figure 5.2: Initial FTTree

The *minimum support* obtained from FT tree is 3. The pruned FT tree will not contain *Iphon*, *Mac* and *Tune* termsets. The resulting FT tree after pruning is shown in Figure 5.3. The closed frequent termsets identified from FT tree shown in Figure 5.3, are *osmosi*, *molecul* and *Appl* each with support 3, and *Water* with support 4. These closed itemsets are sorted in the descending order of support

Table 5.10: Document Coverage Table

Support	Coverage	Not covered Docs at the support
1	1.0	nil
2	1.0	nil
3	1.0	nil
4	0.5	document1, document2, document5, document6

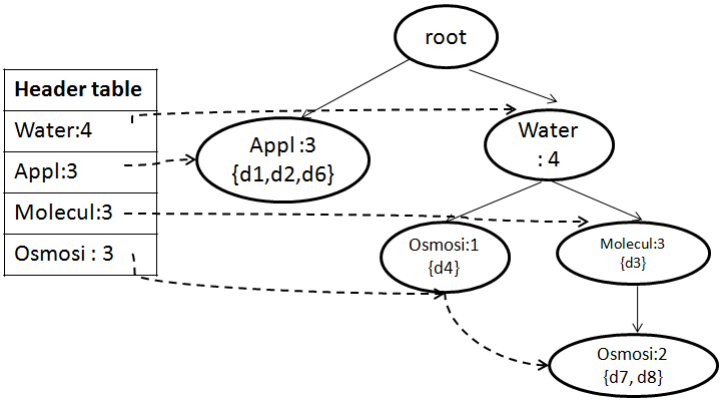

Figure 5.3: Pruned FTTree

resulting in the termsets *water*: 4, *appl*: 3, *molecul*: 3 and *osmosi*: 3 (termsets with same support are sorted in the alphabetic order).

In the clustering process, starting with the itemset *water*, documents 3, 4, 7 and 8 form *cluster 1*. In the growing phase, *molecul* and *osmosi* are added to the list of termsets of this cluster. Next itemset *appl* is taken and the documents 1, 2 and 6 forms *cluster 2*. The incremental process of the clustering method stops here as the closed frequent termsets have exhausted. Now the document 5 has to be classified using Nearest Neighbor classification. The Cosine similarity of document 5 to cluster 1 is 0.0 and to cluster 2 is 0.162 (for this illustration, no predetermined threshold is specified). Hence document 5 gets added to *cluster 2*. During the label generation phase, for *cluster 1*, *water* has support 4, where as *molecul* and *osmosi* have support 3. So *Water* is the label for *cluster 1*. *appl* is the only frequent termset associated with *cluster 2*, hence *Apple* is the label generated for *cluster 2* by unstemming *appl*.

5.3.7 Experiments and Results

The results in Table 5.11 contain the standard parameters used for measuring the clustering performance. The column "AMBIENT Precision" gives the ground truth. The precision of proposed method is near to the ground truth. It is not possible to compute "AMBIENT Recall" for the manually assigned subtopics, as

we are not aware of the number of original relevant results. The Recall of the proposed method tells the fraction of relevant results that are retrieved. Considering the number of results having a manual subtopic assignment as relevant result, Recall tells the number of results retrieved from this number. Fscore, which is a harmonic mean of Precision and Recall, is reported as the last field in the Table 5.11. According to [86], evaluating the clustering performance with comparison to the manually assigned subtopics has some justified differences. One manually assigned subtopic gets divided into small pure groups by the clustering algorithms. Hence the mapping is done between manually assigned subtopics and the clusters identified by the proposed method for calculation of the above measures. This mapping is guided by the Cosine similarity between the centroid vectors (vector space) of manual subtopics and clusters of the method.

Table 5.11: Clusters performance: Precision, Recall and Fscore

S.No	Query	Ambient Precision	Precision	Recall	Fscore
1	Aida	0.6	0.52	0.87	0.65
2	B-52	0.75	0.69	0.92	0.79
3	Beagle	0.85	0.84	0.98	0.9
4	Bronx	0.76	0.73	0.96	0.83
5	Cain	0.38	0.34	0.89	0.5
6	Camel	0.69	0.65	0.94	0.77
7	Coral Sea	0.42	0.36	0.86	0.51
8	Cube	0.48	0.44	0.92	0.6
9	Eos	0.63	0.55	0.87	0.67
10	Excalibur	0.32	0.3	0.94	0.45
11	Fahrenheit	0.66	0.59	0.89	0.71
12	Globe	0.52	0.49	0.94	0.64
13	Hornet	0.44	0.42	0.95	0.58
14	Indigo	0.38	0.37	0.97	0.54
15	Iwo Jima	0.85	0.81	0.95	0.88
16	Jaguar	0.79	0.61	0.77	0.68
17	La Plata	0.66	0.58	0.88	0.7
18	Labyrinth	0.26	0.24	0.92	0.38
19	Landau	0.4	0.36	0.9	0.51
20	Life on Mars	0.83	0.8	0.96	0.87
21	Locust	0.47	0.39	0.83	0.53
22	Magic Mountain	0.41	0.39	0.95	0.55
23	Matador	0.37	0.37	0.97	0.53
24	Metamorphosis	0.54	0.52	0.96	0.68
25	Minotaur	0.51	0.49	0.96	0.65
26	Mira	0.38	0.33	0.87	0.48
27	Mirage	0.34	0.32	0.94	0.48
28	Monte Carlo	0.71	0.67	0.94	0.78
29	Oppenheim	0.41	0.4	0.98	0.57
30	Out of Control	0.18	0.16	0.89	0.27
31	Pelican	0.59	0.54	0.92	0.68
32	Purple Haze	0.27	0.23	0.85	0.36
33	Raam	0.58	0.54	0.93	0.68
34	Rhea	0.52	0.43	0.83	0.57
35	Scorpion	0.44	0.45	1	0.62
36	The Little Mermaid	0.47	0.44	0.94	0.6
37	Tortuga	0.29	0.27	0.9	0.41
38	Urania	0.43	0.39	0.91	0.55
39	Wink	0.46	0.42	0.91	0.58
40	Xanadu	0.48	0.41	0.85	0.55
41	Zebra	0.7	0.65	0.93	0.76
42	Zenith	0.3	0.32	1	0.49
43	Zodiac	0.2	0.2	1	0.33
44	Zombie	0.34	0.32	0.94	0.48

According to [94], Rand Index is a quality measure for clustering algorithms when a gold standard G is given. The Rand index penalizes both false positive and false negative decisions during clustering ([86]). It measures the similarity of the two assignments while ignoring the permutations. It measures whether two objects are in the same clusters in the ground truth clustering and the clustering algorithm. For every pair of data objects whether there is an agreement or disagreement is verified. The Rand index is computed by dividing the number of agreements by the sum of agreements and disagreements. Hence it measures how closely the clustering algorithm matches the ground truth. The proposed Snowball clustering is compared with Lingo, Suffix Tree Clustering and Key SRC (These are reported as the best systems by [11]) and the results are reported in Table 5.12.

Table 5.12: Cluster performance comparison (AMBIENT dataset): Rand index

Clustering method	Rand index
Lingo	62.75
STC	61.48
KeySRC	66.49
Proposed method	62.19

Rand Index values on MORESQUE datasets are given in Table 5.13. As this dataset contains more complex queries when compared to the AMBIENT dataset, the methods performance is lower than AMBIENT. The results for individual queries of MORESQUE are not reported due to space limitations (114 queries).

Table 5.13: Cluster performance comparison (MORESQUE dataset): Rand index

Clustering method	Rand index
Lingo	52.68
STC	51.52
KeySRC	55.52
Proposed method	53.83

The average times taken for the Feature centric GICA on AMBIENT and MORESQUE datasets are tabulated in Table 5.14. These average times include all the times pre processing till display of clusters.

Table 5.14: Time taken for Feature Centric GICA

Dataset	Time taken in milli seconds
AMBIENT	937.18
MORESQUE	1083.72

From the results of Tables 5.12, 5.13 and 5.14 it can be inferred that the

performance of the proposed method is comparable with other popular clustering algorithms that do not use any external resource. The only exception is KeySRC which uses external resource. Other SRC validation measures described in Chapter 2 does not exist for many SRC approaches, hence they are not reported here. Most of those measures are proposed by various authors of SRC algorithms and no common comparison does not exist, except Rand Index.

The time taken for the proposed method is approximately 1000 milliseconds, which can be taken as one second. Hence, it can be inferred that response time of the proposed SRC method is approximately one second.

5.4 Conclusions

The proposed clustering algorithms are following machine learning approach. The parameters used are adaptive, whose values are picked up by the method based on the data. The EM based term pruning resulted in dimensionality reduction without loss of information. The closed frequent itemset analysis complexity is manageable due to the EM based feature selection. The Feature Centric Incremental Clustering is able to produce dynamic clusters with meaningful labels. Further, the clustering method does not require any human interaction or even external knowledge.

Search results are represented by snippet and title. Hence, the proposed Feature centric GICA has been integrated in the MSE lead to search result organization into categorical folders as well as query enhancement. The relevant term set obtained by EM based term set pruning can be employed in search result analysis in Web Mining.

The Object Centric Incremental Clustering is capable of clustering numeric data without the knowledge of the number of clusters.

CHAPTER 6

Consistency Analysis of Query Reformation

A methodology to measure the effectiveness of query reformation is detailed in this chapter. Chi square contingency analysis and Rank correlation are used for this purpose.

6.1 Introduction

Search engine results are highly sensitive to the keywords used in the query. And usually search engine queries are short, with an average length of 2-3 words. Usage of most suitable keywords for the search query is necessary to retrieve more relevant results. Usually there is a very small overlap between the keywords used in the queries and those present in the actual documents. Many users may not have the knowledge of the suitable keywords for their search requirement. When the user is not satisfied with the results of a query, he will follow the reformed queries as suggested by search engines or try his own reformations. It is estimated that approximately 28% of the web search queries are reformations of original queries [100].

Conventional approaches for query reformation make use of terms extracted from WordNet, Top ranked documents and so on. Based on search engine logs, relevance feedback can be captured by observing query and click information and this can be used as pseudo relevance feedback [41]. Few attempts have been made [58], [64], to classify the query reformation techniques and to identify the more commonly attempted and useful reformation techniques. These researchers have also analyzed the conditions in which a query reformation technique is more suitable. But there have been no attempts so far to assess the relevance of the reformed query and its enhancement. The approaches based on pseudo relevance feedback may also include noise. Some reformed queries may be entirely different from users search intention. Few examples where a query substitution may fail to improve relevance are given in [35]. Knowing the relationship between the initial query and the reformed query is useful to improve query reformation strategies as well as improve the way in which the search engines retrieve the results. This

chapter presents the consistency analysis of the initial and reformed queries using statistical measures, to elicit the similarities of web queries and effectiveness of the query enhancement.

6.2 Query Reformation strategies

Search engine log is the main data source for performing query reformation. Using AOL query log, [58] have identified the user sessions and then Initial query and the classes of reformulated queries from the sessions. Using click information present in the log, they have measured the effectiveness of query reformation. Rather than using the whole query as a unit of analysis, [136] has mined the term level associations of terms inside a query. They have proposed context sensitive term substitutions and term additions with the help of probabilistic models by using the query log. A query recommendation is system proposed and evaluated by [78], by making use of the noun, URL, and web community feature spaces.

Query reformation patterns are classified to predict the next query reformation in [63]. They employed an n-gram model to compute the probability of users transitioning from one query reformation state to another, and to determine what order of states provide the best prediction of future states. In [125], a matrix product of the RSV (Retrieval Status Value) vector and the documents-terms matrix is computed and its dual is used to suggest query reformation.

Search engine logs are used by [5, 8, 68, 138] for query reformation. As search engine logs are proprietary, [49] and [35] have come up with query reformation without making use of query logs. First item-clicked feedback measure is shown to be better than pseudo relevance feedback by [49], and have claimed increased query efficiency by testing on TREC Web Track. Reference [35] used anchor text as a substitute for query log. Their results on TREC collection states that query expansion is the safer query reformation technique than substitution and they claimed using anchor text is as effective as search engine logs. Anchor text is used by [91] and [74].

6.3 Prediction of Query Performance

Discounted Cumulative Gain (DCG) and Normalized Discounted Cumulative Gain (NDCG) are measures of effectiveness of a Web search engine algorithm [65]. DCG measures the usefulness, or gain, of a document based on its position in the result list. NDCG is used to compare performance of one query with another. The

Clarity prediction measure [91] is based on measuring the relative entropy between a query language model and its corresponding correlation language model. But as Clarity computation is expensive, many researchers have attempted various techniques for measuring query performance. Visual clues like titles, snippets etc are used in [66] to predict retrieval effectiveness of queries. Relationship between prediction techniques and user based performance measures are investigated in [150]. In [117], the amount of query drift is estimated by measuring the diversity of retrieval scores and claims a better performance on TREC corpora. Instead of using quality measures, in [30], class based statistics which serve as a summary of the document content are used. Rank Time Performance Prediction (RAPP) is proposed in [6], which uses retrieval scores and aggregates of rank time features. Query, result and interaction features are used in [52], and claims usage of these features is better than using only query features.

6.4 Proposed Method

In this section, a framework to evaluate the effectiveness of the reformed queries based on various statistical measures is introduced. All the methods discussed earlier require external tools and depend on complex models. It is felt that a light weight approach is required and hence a framework is proposed. This framework is developed based on query Q and its reformed query Q^R.

This system consists of any general search engine like Google or Yahoo! and query evaluation engine. Given the initial query and a set of reformed queries, the system identifies whether the initial query and reformed queries are similar, reformed query is an enhancement of the original, or is a contradiction of the original. URLs fetched by the query are based on two components: all phrases of the query and individual phrases of the query. The top ranked URLs are from dominating phrases of the query. So identifying the dominating phrases is useful to predict the results of a query.

URLs from the search engine results are retrieved for the initial query (Q) and for the reformed query (Q^R). The association between Q and Q^R is identified by performing Chi square contingency test [139]. A cross table is formed based on the URLs of the queries Q and Q^R as shown in Table 6.1. The Chi square coefficient value is computed as

$$\text{Chi square} = \frac{2 \times N \times (a \times d - b \times c)}{r1 \times r2 \times c1 \times c2} \tag{6.1}$$

The Chi square association analysis is used to assess whether the queries are inde-

Table 6.1: Cross table of the URLs of Q and Q^R

	URLs not present in Q^R results	URLs present in Q^R results	
URLs not present in Q results	*a*	*b*	*r1*
URLs present in Q results	*c*	*d*	*r2*
	c1	c2	N

pendent or dependent. If the Chi square coefficient value is above the significance level (0.05 is popularly used), then the queries can be declared as dependent. Otherwise they are independent. This analysis is useful to know the intention of the search engine in bringing up the results. In case of dependent queries, some phrases are dominated by the phrases of Q. Hence the hidden characteristics of the search engine can be utilized effectively to retrieve the results.

From the URLs present in initial query results and reformed query results, the common URLs are identified. For these common URLs, Rank Correlation coefficients are computed. These values speak about the effectiveness of the reformulated queries. In case of dependent queries, the means of ranks of common URLs of Q and Q^R are computed to know the drift of the ranks. Its value lies between -1 and 1. When it is near to -1, the reformation is a contrast. When it is near to 1, it is more similar, and is not expected to have additional information. When it is close to 0, the queries have independent nature.

6.4.1 Algorithm

The top N results are downloaded from any conventional search engine for the given query and its reformation to be validated. The cross table to compute Chi square is formed based on the presence or absence of the URLs in the set of results. The cross table is used to compute the Chi square value, denoted as θ_1 in the Algorithm 8. The consistent queries satisfy threshold α for their Chi square value. If Chi value is below this threshold the queries are considered to be inconsistent. Rank correlation analysis is performed on the ranks distribution of the URLs that are common for both the queries Q and Q^R. The sign and magnitude of rank correlation analysis is used to decide whether the queries have agreement in same direction or opposite direction.

Algorithm 8: Consistency analysis of Queries

Input : Query Q and Significance level α

Output: Dependent Queries DQ, Independent Queries IDQ, Mean of Q URL ranks as μ_1, Mean of Q^R URL ranks as μ_2

begin

$Q^R \leftarrow$ `QueryReformation` (Q);

$U \leftarrow$ `RankedURLsList` (Q);

$U^R \leftarrow$ `RankedURLsList` (Q^R);

$\theta_1 \leftarrow \chi^2(U, U^R)$;

$\mathsf{DQ} \leftarrow \{Q^R | \theta_1 > \alpha\}$;

$\mathsf{IDQ} \leftarrow \{Q^R | \theta_1 < \alpha\}$;

`// common urls`

$\mathsf{C} \leftarrow U \bigcap U^R$;

$\mathsf{R} \leftarrow \{\texttt{rank}(u) | u \in U \text{ and } u \in \mathsf{C}\}$;

$R^R \leftarrow \{\texttt{rank}(u) | u \in U^R \text{ and } u \in \mathsf{C}\}$;

$\theta_2 \leftarrow \rho_{\mathsf{R},R^R}$;

$\mu_1 \leftarrow$ `mean` (R);

$\mu_2 \leftarrow$ `mean` (R^R);

end

6.4.2 Experiments

Few popular queries and their reformations are submitted to Google search engine and their ranked lists of URLs are captured. These lists are pre-processed to retain only the URLs of web pages, by removing absolute and relative URLs used by the search engine as well as the advertisement URLs. The reformed queries are considered from some of the references discussed in Query Reformation strategies (section 6.2) and those proposed by Google. All these URLs along with rank information are stored in a table. This table is the fundamental data for analysis of query reformation.

Table 6.2: Snapshot of the URLs Table

URL id	URL	Q1	Q2	Q3	Q4
314	http://www.qou.edu/english/scientificResearch/eLearningResearchs/dataMining.pdf	-1	94	-1	-1
315	http://www.mozenda.com/data-mining-software	92	-1	-1	32
316	http://www.twocrows.com/	29	-1	-1	-1
317	http://www.exforsys.com/tutorials/data-mining/information-on-contemporary-data-mining-tools.html	-1	-1	-1	91
318	http://www.dataminingcasestudies.com/	41	-1	-1	-1
319	http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471253847.html	-1	-1	-1	59
320	http://www.spss.com/software/modeling/	-1	-1	-1	65
321	http://dataminingtools.net/blog/	-1	-1	-1	13
322	http://www.dataentryindia.com/data_processing/data_mining.php	16	-1	14	-1
323	http://www.qou.edu/english/scientificResearch/eLearningResearchs/dataMining.pdf	-1	-1	-1	10
324	http://www.mozenda.com/data-mining-software	88	-1	-1	83

In Table 6.2, a snap shot of the URLs table with data mining (Q1) as the initial query, and data mining functionalities (Q2), data mining techniques (Q3), data mining tools (Q4) as the reformed queries is shown. When the URL is not present in search result, its rank is given the value -1. From the URLs table, contingency tables for each pair of Q and Q^R are built and using them, Chi square values are computed. Rank correlation coefficients are computed for the common URL's ranks. The reformed queries which are significant are identified and using them the drift in the mean of the URL's ranks is computed.

6.4.3 Results and Analysis

Table 6.3: Chi Square values

Initial Query=data mining	chi square	P value
data mining software	0.9572	0.3279
data mining articles	0.1998	0.6549
data mining companies	1.1656	0.2803
data mining and privacy	7.9953	0.0047
data mining course	3.1642	0.0753
data mining books	1.4390	0.2303
data mining conference	4.5301	0.0333
data mining pdf	1.1656	0.2803
data mining degree	8.1237	0.0044
data mining techniques	1.9606	0.1615

Table 6.3 shows the Chi square contingency test values with Q = "data mining" and the ten reformed queries as shown in the table as Q^R. As association is computed between Q and each of the reformed query, a 2 X 2 contingency table is used in each case. Hence the degrees of freedom is 1. With degrees of freedom = 1, for 0.05 significance level, the value needed to reject the null hypothesis is 3.841 (using Chi square distribution table). From Table 6.3, "data mining and privacy", "data mining conference" and "data mining degree" values are above the significance level, and the null hypothesis that these queries are independent from Q is rejected. Hence they are correlated with Q and are dependent on Q ("data mining"). Table 6.4 shows the Pearson Rank correlation coefficient [139] values between the query Q = "data mining" and each of the reformed queries. From these values it can be observed that the three significant queries are having the rank correlation value near to 1. When it is +1, it signifies that the common URLs in Q and Q^R are in same order. When it is -1, they are in the opposite order.

For the significant queries in terms of dependency on Q = "data mining", the

Table 6.4: Rank Correlation Coefficients

Queries	$\rho_{URLranks}$
data mining software	0.099
data mining articles	0.251
data mining companies	-0.003
data mining and privacy	1
data mining course	0.981
data mining books	0.673
data mining conference	0.903
data mining pdf	-0.643
data mining degree	-1
data mining techniques	0.719

Table 6.5: Mean Values of URL Ranks

Reformed Query (Q^R)	Mean of the URL's ranks for Q	Mean of the URL's ranks for Q^R
data mining and privacy	45	21
data mining conference	63.25	21
data mining degree	24.5	3.5

mean values of the ranks are shown in Table 6.5. When mean value is reduced in reformed query, it is an enhancement of the original query.

6.5 Conclusions

Consistency analysis of query reformations is peformed with Chi square association analysis and Rank correlation coefficient. The reformed query is associated with the original query when the Chi square value is significant. The reformed query is independent from the original query otherwise. When Chi square value is significant and Rank correlation value is near to 1, the reformed query should be noted as positive enhancement, and its mean rank is smaller than the mean rank of original query. Based on these measures, the reformed query can be considered as an enhancement when Chi square value is significant and Rank correlation is near to 1.

CHAPTER 7

Meta Search Engine

A meta search engine which addresses the query ambiguity caused by polysemantic queries, is built using the tools developed in this thesis. The proposed MSE helps naive user to quickly focus on his/her search interest when the appropriate vocabulary could not be used to frame the initial search query, thus leading to ambiguity. The polysemy nature of the query terms can bring the most popular subtopic results in the first few pages of normal search engines. This makes the user to sift through the flat ranked long list of search results, searching for the result that meets his/her information requirement. But, usually users check one or two pages of search results and give up. If the user can learn the appropriate terms from first few pages, user can reform the query to satisfy his/her information need. With the proposed MSE, user need not have sophisticated learning capability to identify the most suitable subtopic of the polysemantic query. The MSE avoids the fatigue caused by this time consuming activity. The current chapter presents the details of the meta search engine.

7.1 Introduction

Polysemous queries are addressed by various clustering search engines. These are capable of indicating the subtopics of polysemantic queries to resolve query ambiguity. Personalized search engines (e.g. [103]) focus in retrieving search results with weightage to the user profile. These raise privacy concerns and decrease the likelihood of finding new information due to the bias induced by user profile. An interactive search engine based on the current search context, with some smart features like learning about search context, domain, intention and so on, on-the-fly will provide apt search experience. In the current information era, a personalized search engine which is context sensitive and preserves privacy, will meet the aspirations of the web users. Some researchers have made attempts in this direction and brought out tools like SnakeT [44], unfortunately this tool is not in active state at present.

This chapter proposes a prototype of the Meta Search Engine (MSE) based on integration of some of the machine learning/mining tools developed in the previous chapters of the thesis. This MSE has objectives like real time functional-

ity with minimal infrastructure and human interaction feature with user guiding characteristics.

Meta Search Engine which handles Query Ambiguity is the subject matter of this chapter. Whenever a user submits a query, the MSE gathers search results from various Search Engines in the order of 100 to 200. The MSE has the capability to cluster these search results based on the snippet and title of the web pages using the Snowball clustering algorithm (feature centric) developed in this thesis as discussed in chapter 5. Further, models are built for IA based on search result data and cluster label analogous to the techniques developed in chapter 3 of this dissertation. Using these models, IA(s) will be appended leading to dimension enhancement. Using dimension reduction techniques, minimal decision table/system will be arrived as demonstrated in chapter 3 of this thesis. The subsequent search results will be classified among these labeled clusters using classification based on CvGain of chapter 4. Considering each cluster label, the query is modified with an appropriate phrase of each cluster label and search results are extracted. Based on these search results, the impact of query reformation has been measured through the performance evaluation strategies developed in chapter 6. Suggestive reformed queries are arrived based on these impact factors. The visualization and interactive module of the MSE organizes search results in various modes: global rank order, grouped in to folders; with options for subsequent search or query refinement. The exercise of query refinement generates refined query and treats the refined query as a fresh query for MSE and the entire process (fetching search results, clustering, visualization and building the classifier) will be triggered. On the option of subsequent search, subsequent search results are extracted and classified into clusters organized into folders to meet the user requirement. On the option of subsequent URLs, the classifier classifies the search results and posts them into respective folders.

The following section 7.2 deals with design aspects of the proposed MSE. Section 7.3 provides module level illustrations. Section 7.4 contains the demonstrations through snapshots of the MSE. Section 7.5 shows the output of few popular clustering search engines in contrast to the MSE output. The chapter concludes with section 7.6.

7.2 Prototype

The architectural diagram of the MSE is provided in Figure 7.1. Whenever a user submits a query to the MSE, the search results are downloaded from search engine like Google. These are preprocessed by filtering HTML tags, stop-word

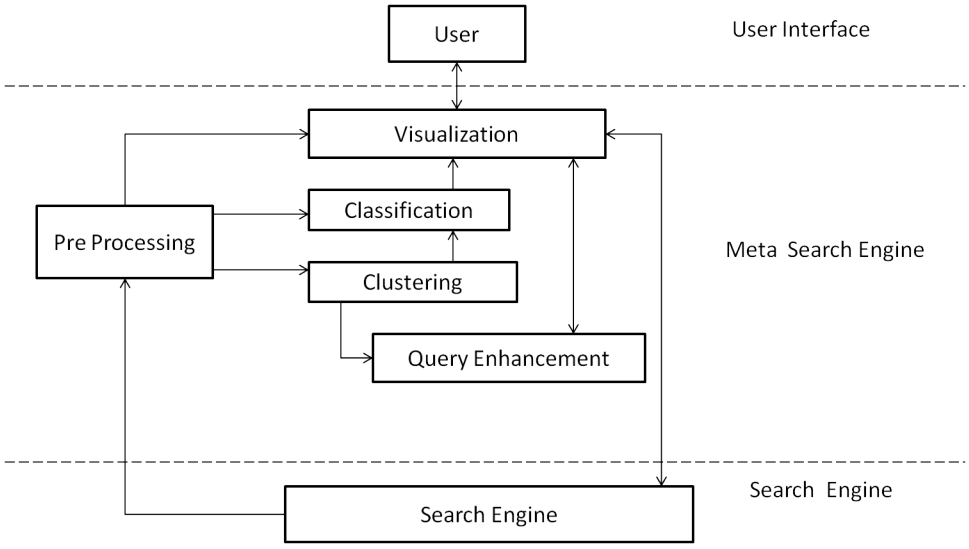

Figure 7.1: Architecture of MSE

removal and stemming. The processed results are clustered using the proposed clustering method. The clusters are displayed with the meaningful labels arrived by the clustering method. The labeled clusters enable the users to narrow down their search to a subtopic of the given ambiguous query. The user is given the provision to download subsequent results constrained on the subtopic of the query and view them classified according to the subtopics relevant to the query. When the user feels that the search results acquired for the query are not sufficient to build the knowledge, subsequent results of the original query are acquired from search engine(s) and classified according to the subtopics relevant to the query. These subsequent results class information is posted to the folders.

The MSE provides query enhancement feature, through which user can refresh and see the results of enhanced query along with its subtopics. The visualization module provides the interface to the user to interact with the MSE.

The context level Data Flow Diagram (DFD) is given in Figure 7.2. It shows the interactions of the MSE with the user and search engine(s). The level 1 DFD is given in Figure 7.3 to explain the details of the MSE.

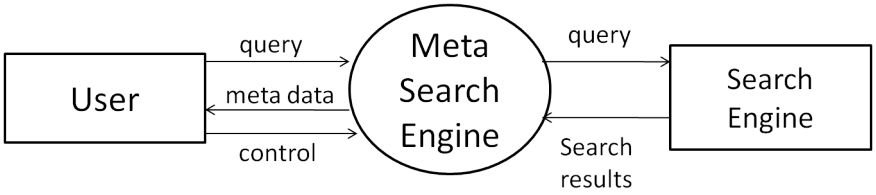

Figure 7.2: Context Level DFD of MSE

The level 2 DFDs give the details of each of the modules and these are provided

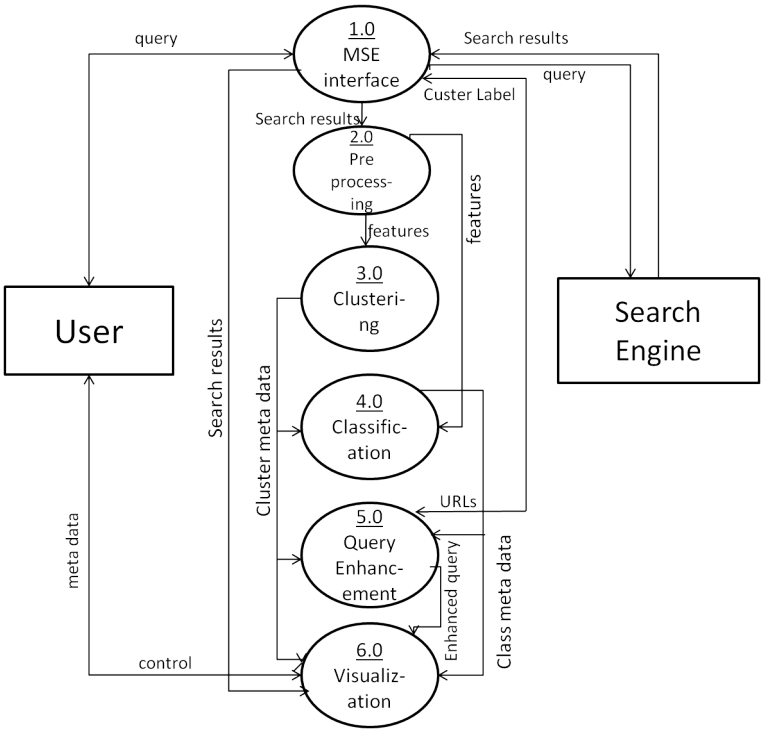

Figure 7.3: Level 1 DFD of MSE

in the following section.

7.3 Module Level Illustrations

7.3.1 Interface to Search Engines

This module sends the query submitted by the user to search engine API to retrieve the search results. The level 2 DFD shows the details of this module and is depicted in Figure 7.4. Most of the search engines provide web search API (e.g. Google

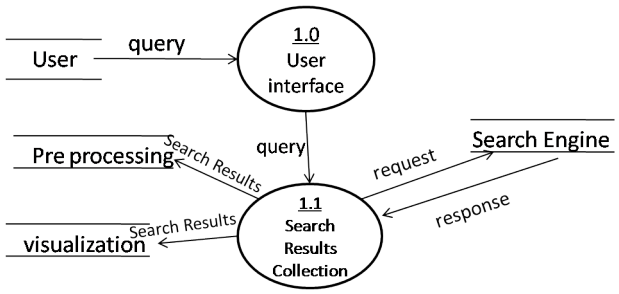

Figure 7.4: MSE Interface

Custom Search, Bing API, Yahoo! BOSS API) to download the search results. Some of search engines offer free web search API while some of them charge with

various tariffs. Each search result consists of four fields: Rank of the result, title of the web page of the result, URL of the result and a short description of the result (snippet). The APIs have the provision to supply the starting rank of the results and the number of results required. Some of the APIs allow only ten results at a time (hence need multiple requests to the search engine) and free services have a limit on the number of queries per day. The top five results obtained through Google Custom Search API are shown in Table 7.1.

Table 7.1: Search Results Sample

Rank	URL	Title	Snippet
1	http://www.java.com/	java.com: Java + You	Get the latest Java Software and explore how Java technology provides a better digital experience.
2	http://en.wikipedia.org/wiki/Java_(programming_language)	Java (programming language) - Wikipedia, the free encyclopedia	Java is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few ...
3	http://www.java.com/ getjava/	Download Free Java Software	This page is your source to download or update your existing Java Runtime Environment, also known as the Java Virtual Machine (JVM, VM, and Java VM), the ...
4	http://www.oracle.com/technetwork/java/index.html	Oracle Technology Network for Java Developers	Feb 21, 2013 ... Beta testing for part one of the Java EE 6 Enterprise Architect Certified Master exam is now underway. Learn how you can save big during the ...
5	http://docs.oracle.com/javase/tutorial/	The Java™Tutorials	Tutorials and reference guides for the Java Programming ...

7.3.2 Preprocessing and Feature Selection

The standard text preprocessing techniques like stop-word removal and stemming [104] are applied on the search results initially. By using Vector space model, each result is represented as a TF-IDF vector according to [86]. EM based feature selection method (discussed in chapter 5) is used to select relevant features. The DFD of the current module (Preprocessing and Feature Selection) is shown in Figure 7.5. The threshold for TF-IDF is obtained as 1.1987, using the EM based feature selection method (detailed in Chapter 5).

7.3.3 Clustering the search results

Using the relevant features, closed frequent itemset analysis is performed to drive the concept formation of the clusters. The FT tree constructed and the minimum support threshold obtained is 4.

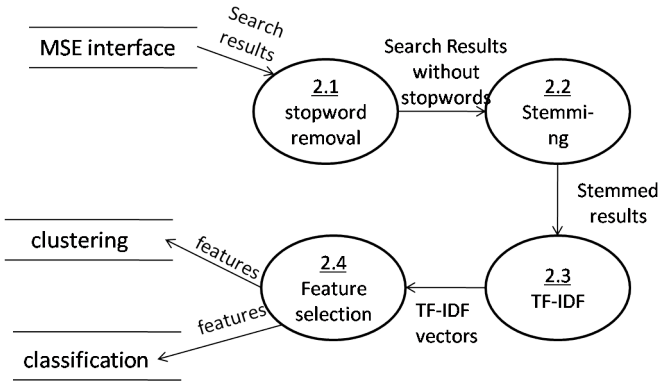

Figure 7.5: Preprocessing and Feature Selection

The closed frequent itemsets obtained along with their corresponding frequencies are

1. 'dog' with frequency 9
2. 'inform' with frequency 8
3. 'breed', 'club', 'dog puppi', 'link', 'linux', 'mar', 'site' and 'websit' with frequency 5
4. 'breed dog', 'dedic', 'free', 'gener', 'group', 'hunt', 'includ', 'kennel', 'offici', 'page', 'person', 'rescu', 'research', 'resourc', 'search', 'small', 'standard', and 'web' with frequency 4

The Feature centric clustering algorithm developed in chapter 5 is used to identify the subtopics of the query. Each cluster will have a set of closed frequent itemsets that form the concept for the cluster. Out of these closed frequent itemsets, the most dominating one is picked up and it is converted to human readable form by adding stopwords and mapping to the unstemmed terms. The DFD detailing this process is shown in Figure 7.6. The clusters obtained are shown below with the cluster label and the number of search results in each cluster:

1. "dog" with 14 results
2. "Mars" with 6 results
3. "information" with 6 results
4. "research" with 5 results
5. "hunting" with 5 results
6. "Linux" with 5 results

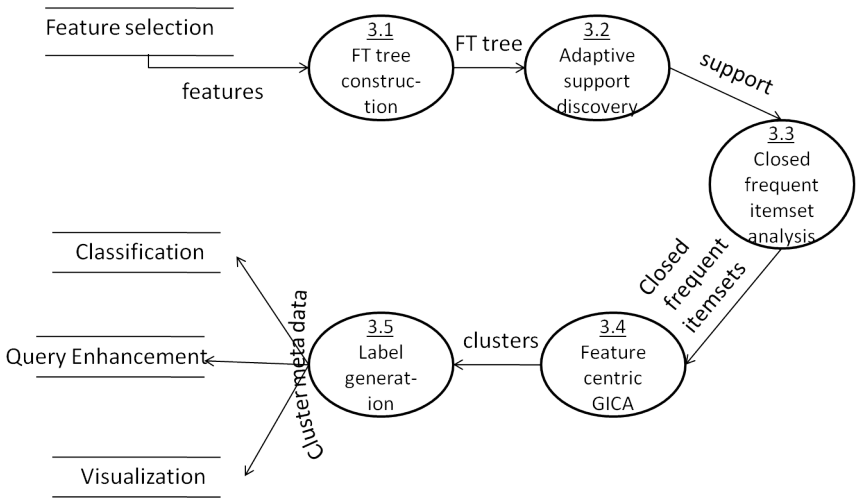

Figure 7.6: DFD for the Clustering Process

7. “links” with 5 results
8. “personal” with 3 results
9. “sites” with 3 results
10. “generally” with 3 results
11. “dedicated” with 3 results
12. “clubs” with 2 results
13. “rescue” with 2 results
14. “clubs” with 2 results
15. “includes” with 2 results
16. “page” with 2 results
17. “website” with 2 results
18. “OTHER” with 9 results

7.3.4 Classification of the search results

The Hybrid classifier (presented in Chapter 3) using CvDT (discussed in Chapter 4) is used for the classification of search results. When the user cannot resolve the ambiguity with the clusters and their contents, then user can utilize this module for further analysis. Based on the user interaction, the subsequent results are downloaded from the Search Engine. These are classified into the subtopics

identified by the clustering algorithm. The DFD of this process is shown in Figure 7.7. With classification of twenty subsequent results, the updated clusters are

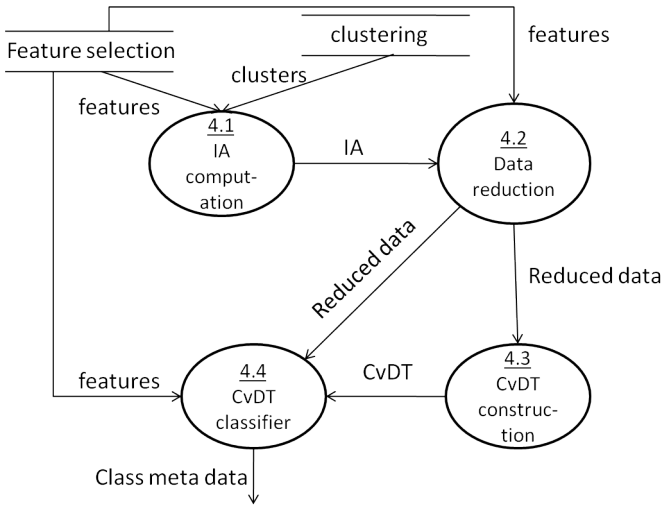

Figure 7.7: Classification Process

shown below:

1. “dog” with 21 results
2. “Mars” with 8 results
3. “information” with 7 results
4. “research” with 5 results
5. “hunting” with 5 results
6. “Linux” with 6 results
7. “links” with 6 results
8. “personal” with 3 results
9. “clubs” with 3 results
10. “sites” with 3 results
11. “generally” with 3 results
12. “dedicated” with 3 results
13. “rescue” with 3 results
14. “clubs” with 2 results

15. “includes” with 4 results
16. “page” with 3 results
17. “website” with 2 results
18. “OTHER” with 12 results

7.3.5 Query Enhancement

Based on the user exploration, the base query used for search is enhanced by using subtopic of interest for the user. The enhanced query is obtained by appending/modifying the base query through the cluster label of user choice. The results of the enhanced query are gathered from search engines’ APIs, clustered and presented to the user. The DFD shown in Figure 7.8 illustrates this module.

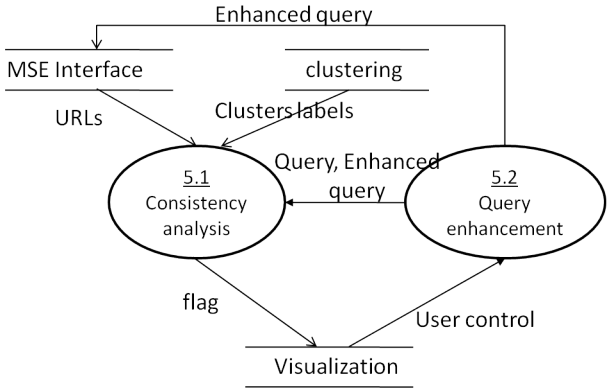

Figure 7.8: Query Enhancement Process

7.3.6 Visualization and User Interface

The clusters are displayed in folder tree method which is most familiar to all users. Once the user search for the query, the displayed window is divided into three regions: 1. horizontal query search space at the top, 2. left pane where the clusters information is displayed and 3. right pane where search results with snippets are. The MSE displays the clusters labels along with the count of the results that form the cluster on the left pane and all the results gathered from search engine(s) on the right pane. Whenever the user wants to explore a subtopic, the corresponding cluster’s results are displayed in the right region to facilitated exploration of the cluster contents. Beside each cluster label, an option to get ‘more’ results of the cluster is provided. This enables the user to gather search results related to the cluster topic and see them after they are classified into

the clusters which are already formed. A provision for retrieving the results of enhanced query is provided, which gets triggered by the user interaction. The DFD of this module illustrated in Figure 7.9.

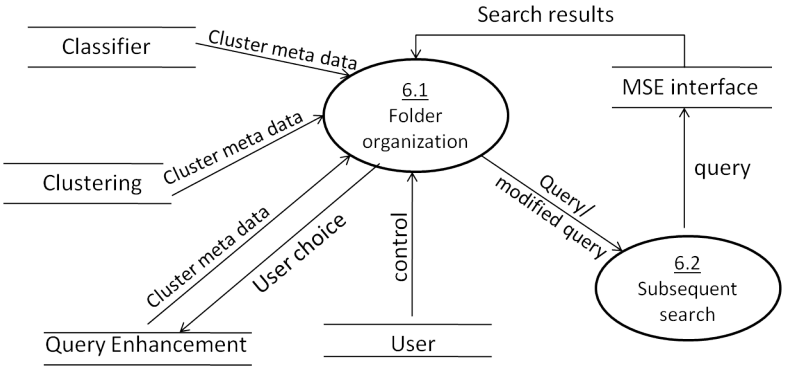

Figure 7.9: Visualization Process

7.4 Demonstration

The input query is shown in Figure 7.10 for which the search results clusters are shown in Figure 7.11. In Figure 7.11, the clusters are shown in the left side. The

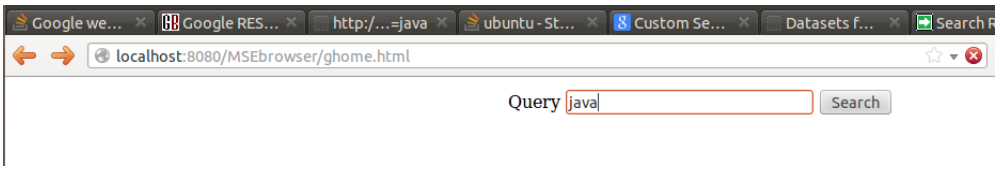

Figure 7.10: Input Query

number of results grouped into each cluster are shown beside each cluster. The right side contains all the results acquired initially (80 results in this demonstration). By clicking on a cluster (which is equivalent to selecting a topic of the query), the cluster contents are displayed on the right side of the screen. Figure 7.12 shows the results of the cluster *Programming*. When *'more'* associated with a cluster is clicked, subsequent results are fetched from search engine. The newly fetched results of the query conditioned on the cluster selected, are classified according to the clusters. The appended clusters with these subsequent results are shown in Figure 7.13. Clicking on a cluster will populate the enhanced query with cluster label. The *Refresh* button is used to obtain **Enhanced Query** which is the reformation of the initial query. A fresh query is formed with this reformation by clicking on the *Refresh* button. The procedure is available in Figure 7.14, whose results are shown in Figure 7.15.

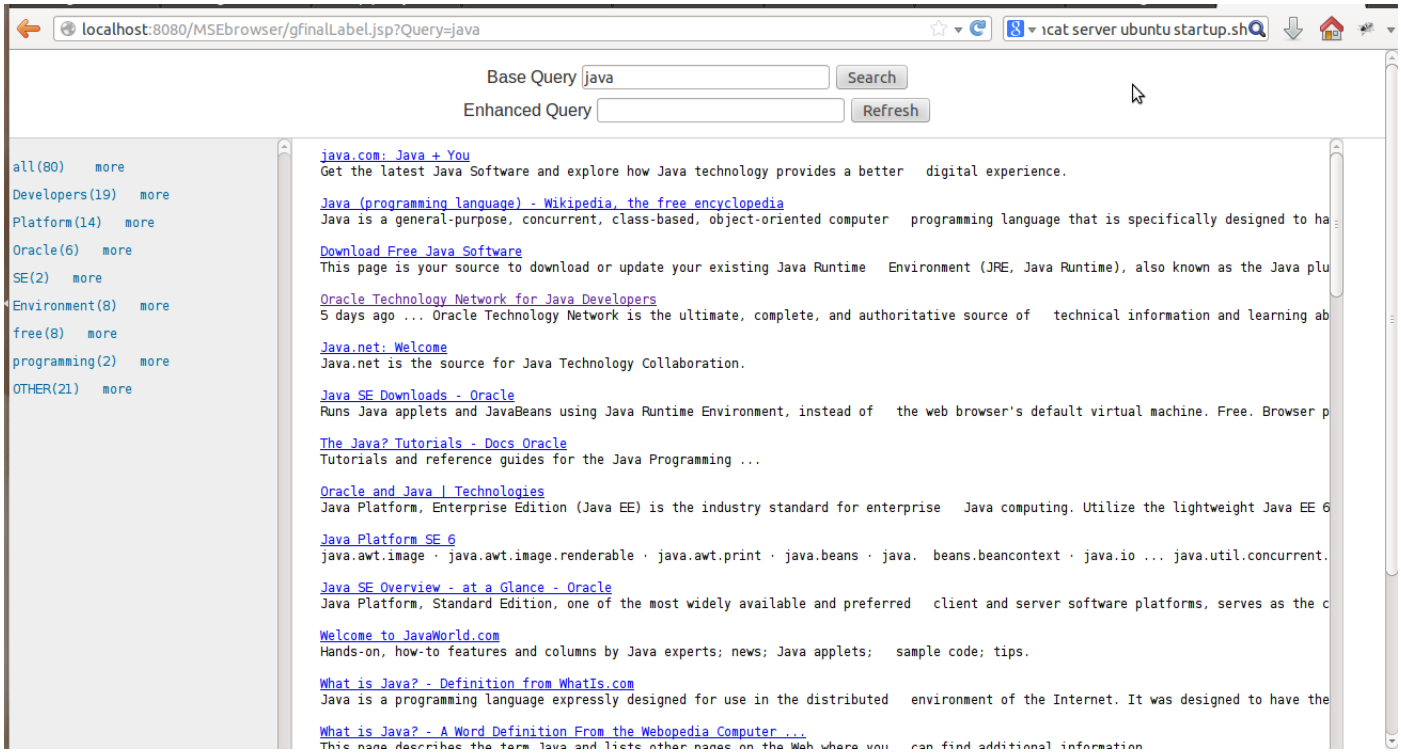

Figure 7.11: Query Results

7.5 Comparison with Existing Clustering Engines

The clusters formed with the proposed MSE are compared with those of few popular clustering algorithms. The clusters identified for the query "apple" with the MSE, "iBoogie", "carrot" and "yippy" are shown in figures 7.16, 7.17, 7.18 and 7.19 respectively.

Figure 7.12: Topic Selection

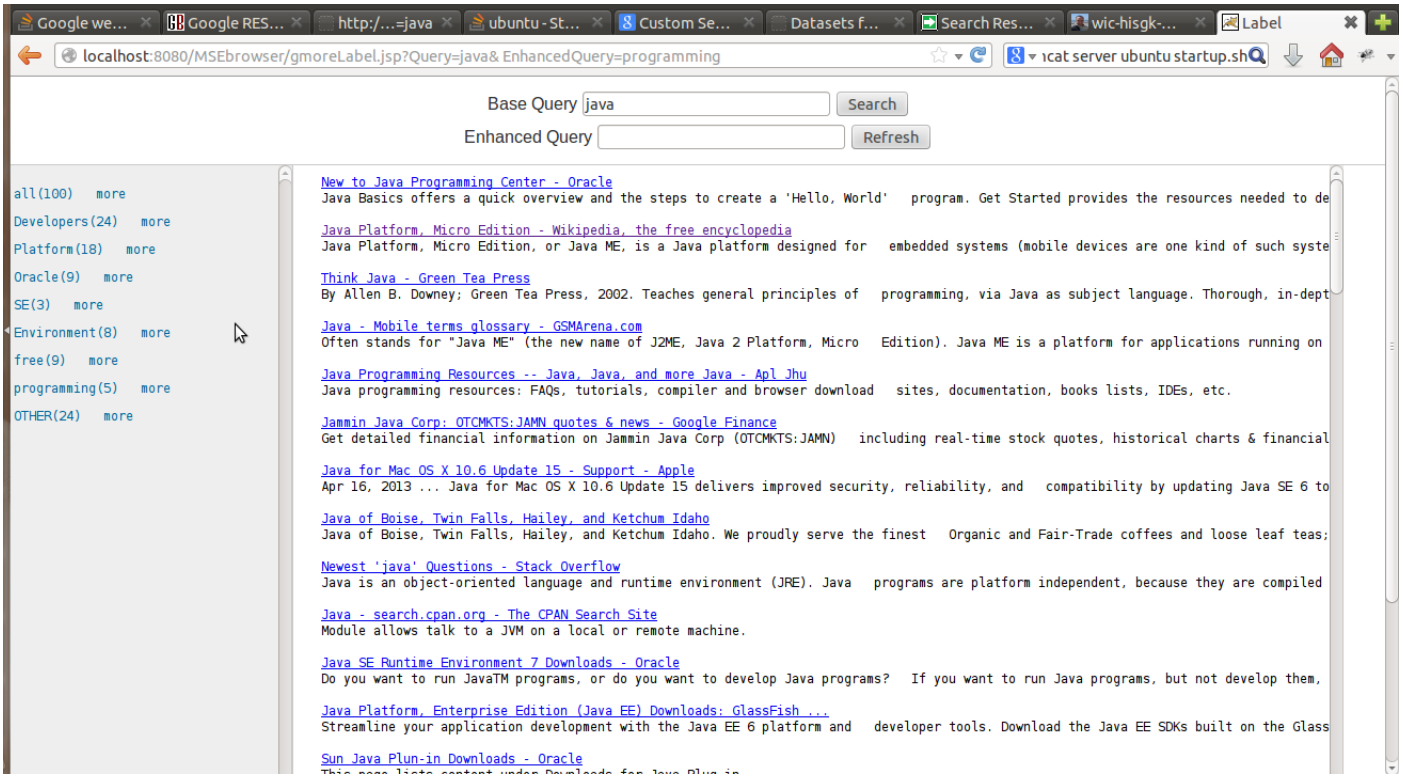

Figure 7.13: Subsequent Results

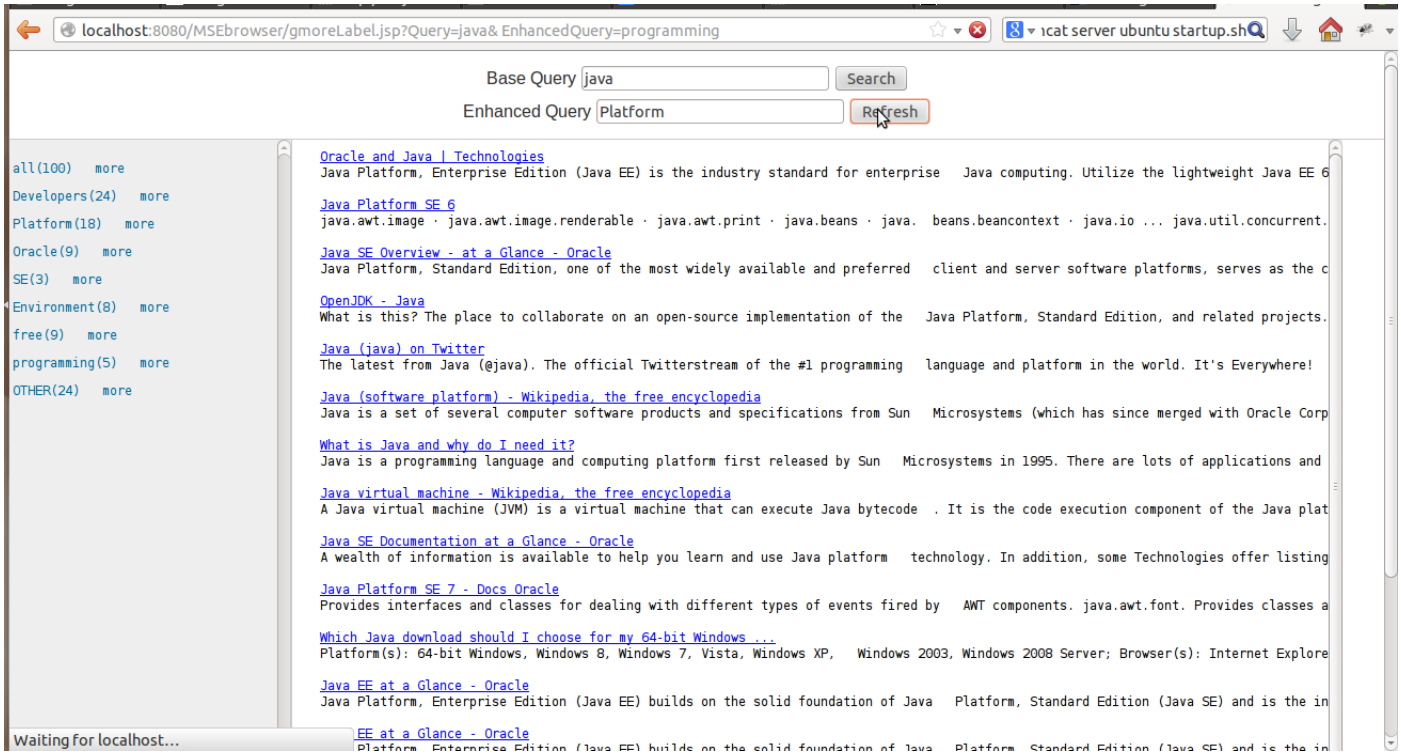

Figure 7.14: Selecting Enhanced Query

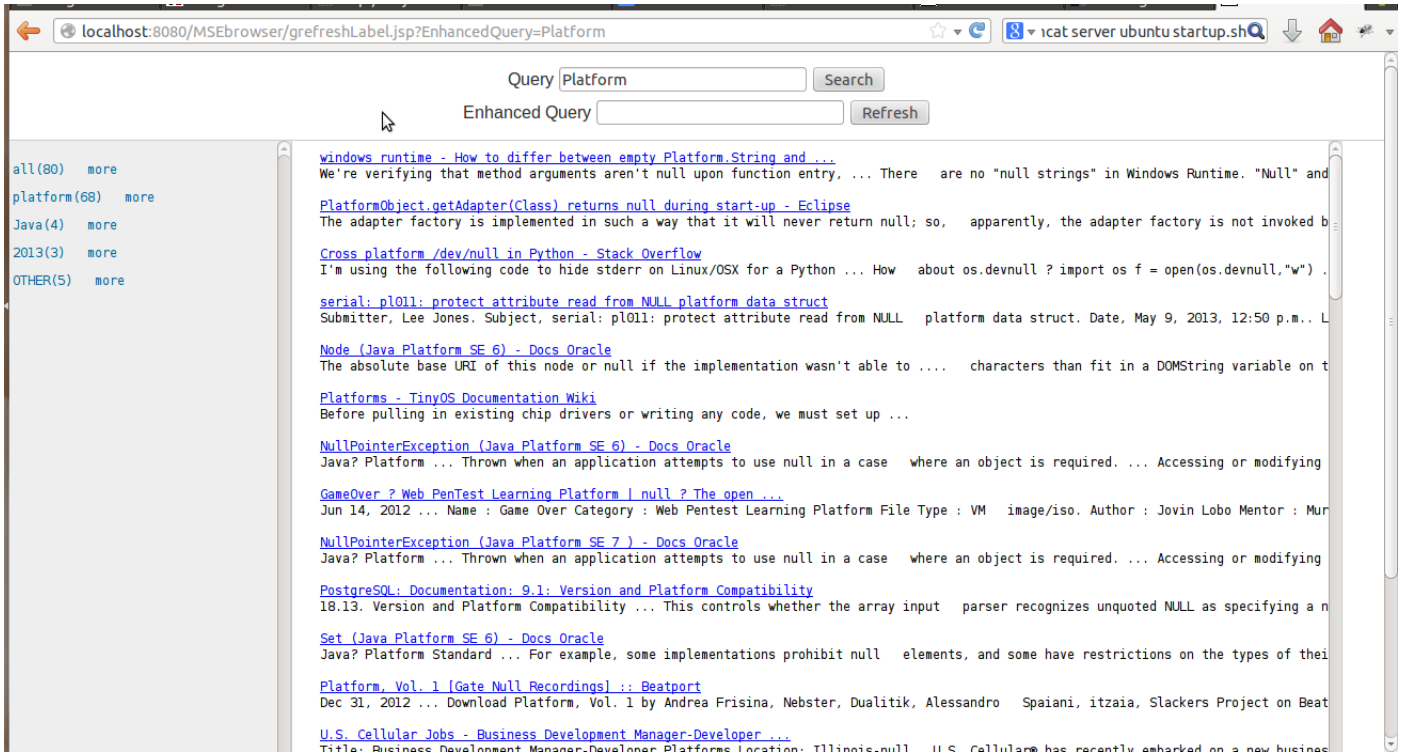

Figure 7.15: Enhanced Query Results

Figure 7.16: MSE results

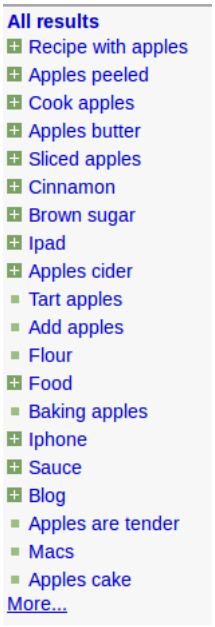

Figure 7.17: iBoogie results

Figure 7.18: Carrot2 results

Figure 7.19: Yippy results

7.6 Conclusions

The proposed MSE identifies the subtopics related to the query posed by the user and provides the interface to explore various subtopics related to the query. This facilitates the user to quickly narrow down the search to his/her topic of interest by selecting the folder (cluster label). When the user feels that the results of a cluster are not sufficient, then the choice of subsequent results through 'more' option will be useful. This option gathers subsequent results from the search engine(s) and classifies according to the clusters formed initially. The proposed MSE also allows the user to refresh the search results through enhanced query, where in, fresh clusters for the enhanced query are made available.

CHAPTER 8

Conclusions

A Meta Search Engine (MSE) is developed in this thesis to address query ambiguity problem. The MSE is an integration of the machine learning tools developed as part of the research. Hence, user interaction is only for query input, selection of subtopic for subtopic exploration and query enhancement. The MSE is capable to organize the search results into folders according to the labels of the clusters, which are arrived by the labeling mechanism based on the content of the cluster. These folders reflect the nature of polysemantic queries.

The clustering method developed is capable to cluster the search results and generate meaningful cluster labels. The classifier designed is used to gather subsequent search results and classify them according to the clusters, to give enough knowledge about the query and subtopics of the query. The query enhancement is guided by the cluster labels and the consistency analysis method demonstrated in this thesis.

A feature selection method is developed based on Expectation Maximization to promote the clustering and classification methods. The thresholds used in the developed tools are arrived based on the data, adaptively without human intervention.

The proposed MSE offers the following benefits:

1. The MSE has the potential to provide better insight as well as enough diversity to take apt decisions, to focus on the subtopic of user interest for polysemous queries.

2. User is free from the burden of learning the subtopics of polysemantic queries. The energies of human can be driven towards his focused area rather than learning from web pages.

3. The current approach followed by search engines is linear list of ranked search results. Through the proposed MSE, it becomes non-linear, which allows the user to jump to a topic.

4. All the search results of a subtopic are grouped and presented as a folder. This enables the user to get enough knowledge about a topic. Further,

through subsequent search/query reformation, the user can gather adequate information to satisfy his/her information need.

5. The time taken by the MSE to facilitate any feature is very little (on the order of a second). Thus the MSE has negligible latency / response time.

The hybrid classifier developed in this thesis produces shorter rules with out loosing the accuracy. This feature is possible through the effective dimensionality reduction due to the meaningful dimension enhancement. Time required for classification using this approach turned out to be significantly low.

The criterion for splitting a node in a decision tree with the the proposed CvGain measure yielded decision trees with low computational cost. Hence, CvDT is suitable for agent based and on line applications.

Greedy incremental clustering method developed in this thesis is demonstrated to form the clusters without the knowledge of the number of clusters. The clustering algorithm follows machine learning approach, the parameters used are adaptive, whose values are picked up by the method based on the data. The clustering algorithm is designed to work for numeric data (object centric) and text data (feature centric).

It is recommended that the tools developed: Snow ball clustering method, Hybrid classifier and CvDT serve as data mining tools based on the validations carried out on the benchmark datasets. The soft computing solutions with these tools are expected to have an edge over the traditional approaches.

Future scope

The event log history of the user at the snapshot of using the MSE can be utilized to the machine learning module considering as relevance feedback for further enhancement of the performance of the MSE. External resources like Wikipedia, Wordnet, ODP etc can be integrated in MSE like other SRC engines.

Problems like XOR that are hard to be modeled by decision trees can be addressed by implicit IA and appropriate IA with non linear forms.

As CvDT has low model building time, it can be extended to various paradigms like distributed and parallel, it can be evaluated by considering the philosophy given in Section 4.8.4 of chapter 4. The distributed construction of CvDT through Map Reduce can bring out the scalable aspects of CvDT. Adaptability and the Limitations of CvDT in various paradigms need thorough investigation.

REFERENCES

[1] Why a Search Engine Might Cluster Concepts to Improve Search Results. http://www.seobythesea.com/2011/01/why-a-search-engine-might-cluster-concepts-to-improve-search-results/.

[2] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 5–14, New York, NY, USA, 2009. ACM.

[3] M. S. Wajahat Ali, C. Raghavendra Rao, C. Bhagvati, and B. L. Deekshatulu. Emtrain++: Em based incremental training algorithm for high accuracy printed character recognition system. *International Journal of Computational Intelligence Research*, 5(4):365–371, 2010.

[4] Georgios Atsaros, Diomidis Spinellis, and Panagiotis Louridas. Site-specific versus general purpose web search engines: A comparative evaluation. In Stefanos Gritzalis, Dimitris Plexousakis, and Dionysios Pnevmatikatos, editors, *PCI 2008: 12th Panhellenic Conference on Informatics*, pages 44–48, Los Alamitos, CA, August 2008. IEEE Computer Society.

[5] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *Proceedings of the 2004 international conference on Current Trends in Database Technology*, EDBT'04, pages 588–596, Berlin, Heidelberg, 2004. Springer-Verlag.

[6] Niranjan Balasubramanian, Giridhar Kumaran, and Vitor R. Carvalho. Predicting query performance on the web. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 785–786, New York, NY, USA, 2010. ACM.

[7] Jan G. Bazan and Marcin S. Szczuka. RSES and RSESlib - a collection of tools for rough set computations. In *Revised Papers from the Second International Conference on Rough Sets and Current Trends in Computing*, RSCTC '00, pages 106–113, London, UK, UK, 2001. Springer-Verlag.

[8] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *In Proceedings of the sixth ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, pages 407–416. Acm Press, 2000.

[9] Florian Beil, Martin Ester, and Xiaowei Xu. Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 436–442, New York, NY, USA, 2002. ACM.

[10] Steven M. Beitzel, Eric C. Jensen, Ophir Frieder, David Grossman, David D. Lewis, Abdur Chowdhury, and Aleksandr Kolcz. Automatic web query classification using labeled and unlabeled training data. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 581–582, New York, NY, USA, 2005. ACM.

[11] Andrea Bernardini, Claudio Carpineto, and Massimiliano D'Amico. Full-subtopic retrieval with keyphrase-based search results clustering. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, pages 206–213, Washington, DC, USA, 2009. IEEE Computer Society.

[12] Ian F. Blake. *An Introduction to Applied Probability*. John Wiley and Sons Inc, 1979.

[13] Thorsten Brants and Alex Franz. Web 1T 5-gram version 1. Linguistic Data Consortium, Philadelphia, 2006. LDC2006T13.

[14] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[15] Leonard A. Breslow and David W. Aha. Simplifying decision trees: A survey. *Knowl. Eng. Rev.*, 12(1):1–40, January 1997.

[16] Peter Bruza, Robert Mcarthur, and Simon Dennis. Interactive internet search: Keyword, directory and query reformulation mechanisms compared. In *In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 280–287. ACM Press, 2000.

[17] Martin D. Buhmann and M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, New York, NY, USA, 2003.

[18] Ana Margarida de Jesus Cardoso Cachopo. *Improving Methods for Single-label Text Categorization*. PhD thesis, Universidade Técnica de LisboaINESC-ID, Portugal, 2007.

[19] Doina Caragea, Adrian Silvescu, and Vasant Honavar. Decision tree induction from distributed heterogeneous autonomous data sources. In *In Proceedings of the Conference on Intelligent Systems Design and Applications (ISDA 03*, pages 341–350. Springer Verlag, 2003.

[20] Claudio Carpineto, Stefano Mizzaro, Giovanni Romano, and Matteo Snidero. Mobile information retrieval with search results clustering: Prototypes and evaluations. *J. Am. Soc. Inf. Sci. Technol.*, 60(5):877–895, May 2009.

[21] Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):17:1–17:38, July 2009.

[22] Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):17:1–17:38, July 2009.

[23] Claudio Carpineto and Giovanni Romano. Exploiting the potential of concept lattices for information retrieval with credo. *Journal of Universal Computer Science*, 10(8):985–1013, aug 2004. |http://www.jucs.org/jucs_10_8/exploiting_the_potential_of.

[24] Claudio Carpineto and Giovanni Romano. Optimal meta search results clustering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 170–177, New York, NY, USA, 2010. ACM.

[25] Hao Chen and Susan Dumais. Bringing order to the web: automatically categorizing search results. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '00, pages 145–152, New York, NY, USA, 2000. ACM.

[26] David Cheng, Ravi Kannan, Santosh Vempala, and Grant Wang. A divide-and-merge methodology for clustering. In *ACM Transactions on Database Systems*, pages 196–205. ACM Press, 2005.

[27] Alexios Chouchoulas and Qiang Shen. Rough set-aided keyword reduction for text categorization. *Applied Artificial Intelligence*, 15(9):843–873, 2001.

[28] CMU Text Learning Group - School of Computer Science. www.cs.cmu.edu/~TextLearning/.

[29] CMU World Wide Knowledge Base (WebKB) project. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/index.html.

[30] Kevyn Collins-Thompson and Paul N. Bennett. Predicting query performance via classification. In *Proceedings of the 32nd European conference on Advances in Information Retrieval*, ECIR'2010, pages 140–152, Berlin, Heidelberg, 2010. Springer-Verlag.

[31] crowdflower.com interface to AMT. https://crowdflower.com/.

[32] KDD cup 2005: Internet user search query categorization. http://www.kdd.org/kdd-cup-2005-internet-user-search-query-categorization.

[33] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. pages 318–329, 1992.

[34] Moghaddam R. Tavakkoli Damghani K.Khalili, Taghavifard M.T. Decision making under uncertain and risky situations. 2009.

[35] Van Dang and Bruce W. Croft. Query reformulation using anchor text. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 41–50, New York, NY, USA, 2010. ACM.

[36] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[37] Antonio Di Marco and Roberto Navigli. Clustering web search results with maximum spanning trees. In *Proceedings of the 12th international conference on Artificial intelligence around man and beyond*, AI*IA'11, pages 201–212, Berlin, Heidelberg, 2011. Springer-Verlag.

[38] Baoshi Ding, Yongqing Zheng, and Shaoyu Zang. A new decision tree algorithm based on rough set theory. In *Proceedings of the 2009 Asia-Pacific Conference on Information Processing - Volume 02*, APCIP '09, pages 326–329, Washington, DC, USA, 2009. IEEE Computer Society.

[39] Yahoo! Directory. http://dir.yahoo.com/.

[40] J. Marin E. B. Hunt and P. J. Stone. Experiments in induction. 1966.

[41] N.E. Efthimiadis. Query expansion. *Annual Review of Information Systems and Technology*, 31:121–187, 1996.

[42] Brian S. Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*. Wiley Publishing, 4th edition, 2009.

[43] Usama M. Fayyad and Keki B. Irani. The attribute selection problem in decision tree generation. In *Proceedings of the tenth national conference on Artificial intelligence*, AAAI'92, pages 104–110. AAAI Press, 1992.

[44] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. *Softw. Pract. Exper.*, 38(2):189–225, February 2008.

[45] Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1625–1628, New York, NY, USA, 2010. ACM.

[46] Benjamin C.M. Fung, Ke Wang, and Martin Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of SIAM international conference on data mining 2003 (SDM 2003*, pages 59–70, 2003.

[47] Fatih Gelgi, Hasan Davulcu, and Srinivas Vadrevu. Term ranking for clustering web search results. In *WebDB*, 2007.

[48] Chris Giannella, Kun Liu, Todd Olsen, and Hillol Kargupta. Communication efficient construction of decision trees over heterogeneously distributed data. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '04, pages 67–74, Washington, DC, USA, 2004. IEEE Computer Society.

[49] Yu-Xi Gong, Rong Luo, and Min-Xia Zhang. Query reformulation based on improved pseudo feedback. In *Proceedings of the 2009 First International Workshop on Education Technology and Computer Science - Volume 02*, ETCS '09, pages 269–271, Washington, DC, USA, 2009. IEEE Computer Society.

[50] Cristina González-Caro and Ricardo Baeza-Yates. A multi-faceted approach to query intent classification. In *Proceedings of the 18th international conference on String processing and information retrieval*, SPIRE'11, pages 368–379, Berlin, Heidelberg, 2011. Springer-Verlag.

[51] Luis Gonzlez Abril and Francisco Velasco Morente. The similarity between the square of the coefficient of variation and the gini index of a general random variable = similitud entre el cuadrado del coeficiente de variacin y el ndice de gini en un. *Revista de Mtodos Cuantitativos para la Economa y la Empresa = Journal of Quantitative Methods for Economics and Business Administration*, 10(1):5–18, December 2010.

[52] Qi Guo, Ryen W. White, Susan T. Dumais, Jue Wang, and Blake Anderson. Predicting query performance using query, result, and user interaction features. In *Adaptivity, Personalization and Fusion of Heterogeneous Information*, RIAO '10, pages 198–201, Paris, France, France, 2010. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

[53] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[54] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.

[55] David Heath, Simon Kasif, and Steven Salzberg. Induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2(2):1–32, 1993.

[56] Reza Taghizadeh Hemayati, Weiyi Meng, and Clement Yu. Identifying and ranking possible semantic and common usage categories of search engine queries. In *Proceedings of the 11th international conference on Web information systems engineering*, WISE'10, pages 254–261, Berlin, Heidelberg, 2010. Springer-Verlag.

[57] Andreas Hotho, Steffen Staab, and Gerd Stumme. Explaining text clustering results using semantic structures. In *In Principles of Data Mining and Knowledge Discovery, 7th European Conference, PKDD 2003*, pages 217–228. Springer, 2003.

[58] Jeff Huang and Efthimis N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 77–86, New York, NY, USA, 2009. ACM.

[59] E. B. Hunt, J. Marin, and P. J. Stone. *Experiments in induction*. Academic Press, New York, NY, 1966.

[60] SMART information retrieval system at Cornell University. http://www.lextek.com/manuals/onix/stopwords2.html.

[61] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[62] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. Determining the informational, navigational, and transactional intent of web queries. *Inf. Process. Manage.*, 44(3):1251–1266, May 2008.

[63] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. Patterns of query reformulation during web searching. *JASIST*, 60(7):1358–1371, 2009.

[64] Bernard J. Jansen, Mimi Zhang, and Amanda Spink. Patterns and transitions of query reformulation during web searching. *IJWIS*, 3(4):328–340, 2007.

[65] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002.

[66] Eric C. Jensen, Steven M. Beitzel, David Grossman, Ophir Frieder, and Abdur Chowdhury. Predicting query difficulty on the web by learning visual clues. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 615–616, New York, NY, USA, 2005. ACM.

[67] S. Jinarat, C. Haruechaiyasak, and A. Rungsawang. Improving web search result categorization using knowledge from web taxonomy. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, volume 02, pages 726–730, 2009.

[68] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 387–396, New York, NY, USA, 2006. ACM.

[69] Rahul Joshi and Y. Alp Aslandogan. Concept-based web search using domain prediction and parallel query expansion. In *IRI*, pages 166–171. IEEE Systems, Man, and Cybernetics Society, 2006.

[70] Mika Käki. Findex: search result categories help users when document ranking fails. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 131–140, New York, NY, USA, 2005. ACM.

[71] Maryam Kamvar and Shumeet Baluja. A large scale study of wireless search behavior: Google mobile search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 701–709, New York, NY, USA, 2006. ACM.

[72] Daniel Karp, Yves Schabes, Martin Zaidel, and Dania Egedi. A freely available wide coverage morphological analyzer for english. In *COLING*, pages 950–955, 1992.

[73] Stella Kopidaki, Panagiotis Papadakos, and Yannis Tzitzikas. Stc+ and nm-stc: Two novel online results clustering methods for web searching. In *Proceedings of the 10th International Conference on Web Information Systems Engineering*, WISE '09, pages 523–537, Berlin, Heidelberg, 2009. Springer-Verlag.

[74] Reiner Kraft and Jason Zien. Mining anchor text for query refinement. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 666–674, New York, NY, USA, 2004. ACM.

[75] Krishna Kummamuru, Rohit Lotlikar, Shourya Roy, Karan Singal, and Raghu Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 658–665, New York, NY, USA, 2004. ACM.

[76] Dawn Lawrie, W. Bruce Croft, and Arnold Rosenberg. Finding topic words for hierarchical summarization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 349–357, New York, NY, USA, 2001. ACM.

[77] Anton Leuski. Evaluating document clustering for interactive information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 33–40, New York, NY, USA, 2001. ACM.

[78] Lin Li, Shingo Otsuka, and Masaru Kitsuregawa. Query recommendation using large-scale web access logs and web page archive. In *Proceedings of the 19th international conference on Database and Expert Systems Applications*, DEXA '08, pages 134–141, Berlin, Heidelberg, 2008. Springer-Verlag.

[79] Tao Liu, Shengping Liu, and Zheng Chen. An evaluation on feature selection for text clustering. In *In ICML*, pages 488–495, 2003.

[80] Ying Liu, Wenyuan Li, Yongjing Lin, and Liping Jing. Spectral geometry for simultaneously clustering and ranking query search results. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 539–546, New York, NY, USA, 2008. ACM.

[81] S. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, September 2006.

[82] Y. S. Maarek, R. Fagin, I. Z. Ben-Shaul, and D. Pelleg. Ephemeral document clustering for web applications. Technical Report RJ 10186, IBM, 2000.

[83] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[84] Hassan H. Malik and John R. Kender. High quality, efficient hierarchical document clustering using closed interesting itemsets. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 991–996, Washington, DC, USA, 2006. IEEE Computer Society.

[85] Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, SODA '90, pages 319–327, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

[86] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[87] Irmina Maslowska and Roman Slowinski. Hierarchical clustering of text corpora using suffix trees. In Mieczyslaw A. Klopotek, Slawomir T. Wierzchon, and Krzysztof Trojanowski, editors, *Intelligent Information Processing and Web Mining, Proceedings of the International IIS: IIPWM 03 Conference held in Zakopane, Poland, June 2-5, 2003*, Advances in Soft Computing, pages 179–188. Springer, 2003.

[88] Ryszard S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20(2):111 – 161, 1983.

[89] S. Minz and R. Jain. Refining decision tree classifiers using rough set tools. *Int. J. Hybrid Intell. Syst.*, 2(2):133–148, April 2005.

[90] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[91] Ramesh Nallapati, Bruce Croft, and James Allan. Relevant query feedback in statistical language modeling. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 560–563, New York, NY, USA, 2003. ACM.

[92] Roberto Navigli. Word sense disambiguation: a survey. *ACM COMPUTING SURVEYS*, 41(2):1–69, 2009.

[93] Roberto Navigli. Semantic is beautiful: Clustering and diversifying search results with graph-based word sense induction. In Giambattista Amati, Claudio Carpineto, and Giovanni Semeraro, editors, *IIR*, volume 835 of *CEUR Workshop Proceedings*, page 1. CEUR-WS.org, 2012.

[94] Roberto Navigli and Giuseppe Crisafulli. Inducing word senses to improve web search result clustering. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 116–126, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[95] Chi Lang Ngo and Hung Son Nguyen. A tolerance rough set approach to clustering web search results. In Jean-Franois Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *PKDD*, volume 3202 of *Lecture Notes in Computer Science*, pages 515–517. Springer, 2004.

[96] most comprehensive human-edited directory of the Web Open Directory Project, the largest. http://www.dmoz.org/.

[97] Stanislaw Osinski and Dawid Weiss. Conceptual clustering using lingo algorithm: Evaluation on open directory project data. In *In IIPWM04*, pages 369–377, 2004.

[98] Stanislaw Osinski and Dawid Weiss. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3):48–54, May 2005.

[99] Web page classification. http://webpageclassification.com/index.pl.

[100] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems*, InfoScale '06, New York, NY, USA, 2006. ACM.

[101] Zdzislaw Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[102] Jian Pei, Jiawei Han, and Runying Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *Proceedings of ACM-SIGMOD International Workshop on Data Mining and Knowledge Discovery*, pages 21–30, Dallas, TX, 2000.

[103] James Pitkow, Hinrich Schütze, Todd Cass, Rob Cooley, Don Turnbull, Andy Edmonds, Eytan Adar, and Thomas Breuel. Personalized search. *Commun. ACM*, 45(9):50–55, September 2002.

[104] M. F. Porter. Readings in information retrieval. chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[105] Politechnika Poznaska, Instytut Informatyki, Dawid Weiss, Dawid Weiss, Rozprawa Doktorska, Wydziau Informatyki Zarz, and Politechniki Poznaskiej. Descriptive clustering as a method for exploring text collections. Technical report, 2006.

[106] Hsiao-Tieh Pu, Sih-Ying Chen, and Pei-Yi Kuo. An empirical evaluation on textual results clustering for web search. *Proceedings of the American Society for Information Science and Technology*, 46(1):1–16, 2009.

[107] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986.

[108] J. R. Quinlan. Learning with continuous classes. pages 343–348. World Scientific, 1992.

[109] J. R. Quinlan. Combining instance-based and model-based learning. pages 236–243. Morgan Kaufmann, 1993.

[110] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[111] Y. Ramadevi and C. Raghavendra Rao. Reduct based decision tree (rdt). *International Journal of Computer Sciences and Engineering Systems*, 2(4), 2008.

[112] William M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[113] a toolkit for analysis of table data Rough Set Exploration System. http://logic.mimuw.edu.pl/~rses/about.html.

[114] Mark Sanderson. Word sense disambiguation and information retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 142–151, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[115] Mark Sanderson. Ambiguous queries: test collections need more sense. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 499–506, New York, NY, USA, 2008. ACM.

[116] Ugo Scaiella, Paolo Ferragina, Andrea Marino, and Massimiliano Ciaramita. Topical clustering of search results. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 223–232, New York, NY, USA, 2012.

[117] Anna Shtok, Oren Kurland, and David Carmel. Predicting query performance by query-drift estimation. In *Proceedings of the 2nd International Conference on Theory of Information Retrieval: Advances in Information Retrieval Theory*, ICTIR '09, pages 305–312, Berlin, Heidelberg, 2009. Springer-Verlag.

[118] The size of the World Wide Web (The Internet). http://www.worldwidewebsize.com/.

[119] George W. Snedecor and William G. Cochran. Statistical methods, Seventh edition. page 508, 1980.

[120] Software and Web User Experience Testing. http://www.techsmith.com/morae.asp.

[121] Ruihua Song, Zhenxiao Luo, Jian-Yun Nie, Yong Yu, and Hsiao-Wuen Hon. Identification of ambiguous queries in web search. *Inf. Process. Manage.*, 45(2):216–229, March 2009.

[122] Myra Spiliopoulou, Markus Schaal, Roland M. Müller, and Marko Brunzel. Evaluation of ontology enhancement tools. In *Proceedings of the 2005 joint international conference on Semantics, Web and Mining*, EWMF'05/KDO'05, pages 132–146, Berlin, Heidelberg, 2006. Springer-Verlag.

[123] Google stops using ODP. http://www.en.wikipedia.org/wiki/Open_Directory_Project#cite_note-21.

[124] Mona Taghavi, Ahmed Patel, Nikita Schmidt, Christopher Wills, and Yiqi Tew. An analysis of web proxy logs with query distribution pattern approach for search engines. *Comput. Stand. Interfaces*, 34(1):162–170, January 2012.

[125] Imen Taktak, Mohamed Tmar, and Abdelmajid Ben Hamadou. Query reformulation based on relevance feedback. In *Proceedings of the 8th International Conference on Flexible Query Answering Systems*, FQAS '09, pages 134–144, Berlin, Heidelberg, 2009. Springer-Verlag.

[126] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[127] Wikipedia:Links to (disambiguation) pages. http://en.wikipedia.org/wiki/Wikipedia:Links_to_%28disambiguation%29_pages.

[128] Ozgur Turetken and Ramesh Sharda. Clustering-based visual interfaces for presentation of web search results: An empirical investigation. *Information Systems Frontiers*, 7(3):273–297, July 2005.

[129] domain theories UCI Machine Learning Repository, a collection of databases and data generators. http://archive.ics.uci.edu/ml/datasets.html.

[130] J. Umakant, K. Sudhakar, P.M. Mujamdar, and C. Raghavendra Rao. Customized regression model for improving low fidelity analysis tool. *American Institute of Aeronautics and Astronautics*, 2006.

[131] J. Umakant, K. Sudhakar, P.M. Mujamdar, and C. Raghavendra Rao. Ranking based uncertainty quantification for a multifidelity design approach. *Journal of Aircraft*, 44(2), 2007.

[132] The 4 Universities Data Set. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/.

[133] İlhan Uysal and H. Altay Güvenir. An overview of regression techniques for knowledge discovery. *Knowl. Eng. Rev.*, 14(4):319–340, December 1999.

[134] Vivismo. http://www-01.ibm.com/software/data/information-optimization/.

[135] Xuanhui Wang and ChengXiang Zhai. Learn from web search logs to organize search results. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 87–94, New York, NY, USA, 2007. ACM.

[136] Xuanhui Wang and ChengXiang Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 479–488, New York, NY, USA, 2008. ACM.

[137] Dawid Weiss and Jerzy Stefanowski. Web search results clustering in polish: Experimental evaluation of carrot. In Mieczyslaw A. Klopotek, Slawomir T. Wierzchon, and Krzysztof Trojanowski, editors, *IIS*, Advances in Soft Computing, pages 209–218. Springer, 2003.

[138] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 162–168, New York, NY, USA, 2001. ACM.

[139] Douglas C. Montgomery Willium W. Hines. *Probability and Statistics in Engineering and Management Science*. John Wiley & Sons, Inc., New York, NY, USA, 3 edition, 2003.

[140] a large lexical database of English WordNet. http://wordnet.princeton.edu/.

[141] Yunjie Xu and Hainan Yin. Novelty and topicality in interactive information retrieval. *J. Am. Soc. Inf. Sci. Technol.*, 59(2):201–215, January 2008.

[142] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[143] Shiren Ye, Tat-Seng Chua, and Jeremy R. Kei. Querying and clustering web pages about persons and organizations. In *Web Intelligence*, pages 344–350. IEEE Computer Society, 2003.

[144] Hwanjo Yu, Duane Searsmith, Xiaolei Li, and Jiawei Han. Scalable construction of topic directory with nonparametric closed termset mining. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '04, pages 563–566, Washington, DC, USA, 2004. IEEE Computer Society.

[145] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. pages 46–54, 1998.

[146] Oren Zamir and Oren Etzioni. Grouper: A dynamic clustering interface to web search results. pages 1361–1374, 1999.

[147] Hua-Jun Zeng, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. Learning to cluster web search results. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 210–217, New York, NY, USA, 2004. ACM.

[148] Cheng Xiang Zhai, William W. Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 10–17, New York, NY, USA, 2003. ACM.

[149] Dell Zhang and Yisheng Dong. Semantic, hierarchical, online clustering of web search results. In *APWeb*, pages 69–78, 2004.

[150] Ying Zhao and Falk Scholer. Predicting query performance for user-based search tasks. In *Proceedings of the Australian Document Computing Symposium*, pages 112–115, Melbourne, Australia, 2007.