

The Pennsylvania State University
The Graduate School

AUTOMATIC TEXT-BASED EXPLANATION OF EVENTS

A Thesis in
Computer Science and Engineering
by
Sandip Debnath

© 2005 Sandip Debnath

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2005

The thesis of Sandip Debnath was reviewed and approved* by the following:

C Lee Giles

David Reese Professor of Information Sciences and Technology and Professor of Computer Science and Technology

Thesis Co-Advisor, Co-Chair of Committee

Prasenjit Mitra

Assistant Professor of Information Sciences and Technology

Thesis Co-Advisor, Co-Chair of Committee

Hongyuan Zha

Professor of Computer Science and Engineering

Tracy Mullen

Assistant Professor of Information Sciences and Technology

Arun Upneja

Associate Professor of School of Hospitality Management

Raj Acharya

Department Head of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

With the abundance of publicly available information on the Web reflecting the ever-changing nature of world events, one question naturally comes to our mind – how can we explain a specific event using this vast amount of information? This enormous information source is freely available but its unstructured nature, and inherent noise demand sophisticated techniques to understand reasons behind any event. Modern day search engines are too general in nature. They only try to find documents which contain the query keyword(s) and supposedly most relevant, but this search method has an inherent limitation. At the end, they are just query matching engines. They do not have and do not require an understanding of the domain knowledge and specialized document processing techniques to provide an explanation of any event. By the word “event” we mean an occurrence to which we can associate a time, e.g. “the IBM stock price went up in the third week of June, 2004”, or “More than 1.5 million people lost electricity in Florida in September 2004”. Newspapers or on-line news articles are full of events like these. Though these articles are important and useful information sources, they are noisy and unstructured compared to structural information sources such as relational databases.

In this thesis, I propose a novel assembly of techniques which can be applied on unstructured information sources such as news articles, news reports, or other text-based documents.

These techniques, applied in a step-by-step fashion, will permit proper analysis of the information source and can provide text-based explanations for major events that have occurred during a given time-frame. Given a noisy, unstructured, weekly-organized source of information, our method can reduce noise with high precision,

sort them (according to specified attributes) with high accuracy, find relevance to the domain of concern with high precision, and at the end can show a text-based explanation (in the form of keywords, phrases or even sentences) for a particular event.

Table of Contents

List of Figures	xi
List of Tables	xv
Acknowledgments	xx
Chapter 1	
Introduction	1
1.1 Motivation	3
1.2 Problem Statement	4
1.2.1 Core Issues	4
1.2.2 Contribution	5
1.2.3 Probable Applications	6
Chapter 2	
Sports Market and Event-Price Correlation	7
2.1 Prelude	7
2.1.1 Brief Description	8
2.2 Introduction	8
2.3 Background	10
2.3.1 Related work	10
2.3.2 Metrics	12
2.4 Soccer World Cup 2002	13
2.4.1 Delay Calculation	14
2.5 2002 NBA Championship	18

2.6	Soccer and Basketball	20
2.7	WSE and the Iowa Electronic Market	22
2.8	Conclusion	23

Chapter 3

	System Design	26
3.1	Description of the Architecture	27
3.1.1	Getting the Information	27
3.1.2	Getting the Right Information: Filtering Out the Noise . . .	27
3.1.3	Right Information at the Right Time: Temporal Ordering .	29
3.1.4	Getting the Relevant Information: Relevance Measure	29
3.1.5	Building Domain Ontology	30
3.1.6	Analyzer	31
3.1.6.1	Keyword-based Model	31
3.1.6.2	Sentence-based Model	31

Chapter 4

	Getting the Right Information: Feature-based Extraction of Informative Blocks from Web-pages	33
4.1	Motivation	34
4.2	Introduction	34
4.3	Related Work	38
4.4	Segmenting Web-pages into Blocks	41
4.4.1	Block Features	42
4.4.2	Inverse Block Document Frequency and Block Similarity . .	43
4.5	Algorithm: ContentExtractor	44
4.5.1	GetBlockSet	44
4.5.2	GetBlocks	44
4.5.3	Identifying Primary Content Blocks	46
4.5.4	ContentExtractor	46
4.5.4.1	The Similarity Function and Threshold	46
4.5.4.2	Complexity Measure	46
4.6	Algorithm: FeatureExtractor	
	/K-FeatureExtractor	49
4.6.1	Block Features	50
4.6.2	K-FeatureExtractor	51
4.7	Experimental Evaluation	53
4.7.1	First Comparison: With the LH Algorithm	53
4.7.2	Metric Used	53
4.7.3	Data Set	55

4.7.4	Performance Comparison	56
4.7.5	b-Precision and b-Recall	57
4.7.6	Execution Time	58
4.7.7	Second Comparison: With Shingling Algorithm	58
4.8	Algorithm: L-Extractor	61
4.9	Conclusions	62

Chapter 5

Right Information at Right Time: Temporal Ordering		65
5.1	Motivation	66
5.2	Introduction	66
5.3	Related Work	68
5.4	Our Approach	70
5.4.1	Example	70
5.5	Time format and Grammar	72
5.6	Data Preparation and Training Phase	74
5.6.1	Data Parameters	75
5.7	Testing and Evaluation Phase	78
5.8	Conclusion	80

Chapter 6

Analyzer Model		81
6.1	Keyword-based Explanation Model	82
6.1.1	Fixed Window Approach	82
6.1.1.1	Definition	83
6.1.2	Variable Window Approach	83
6.1.2.1	Definition	83
6.2	Sentence-based Explanation Model	84

Chapter 7

Keyword-based Model (Fixed Window)		86
7.1	Event Description	87
7.1.1	Market Events	87
7.1.2	Method and Notation	87
7.1.2.1	Logarithmic Score	87
7.1.2.2	Notation	88
7.2	Implementation	89
7.2.1	Finding out Pivotal dates	89
7.2.2	Dates Found	89
7.2.3	Data	90

7.2.4	Our method	90
7.2.4.1	Feature Entropy	91
7.2.4.2	Kullback-Liebler Divergence	93
7.3	Results	94
7.4	Conclusion	96

Chapter 8

Keyword-based Model and Need of an Ontology	97
8.1	Prelude 97
8.2	Real Market Analysis 98
8.2.1	Differences 99
8.3	Evaluation of Keyword-based Model 99
8.3.1	Metric 100
8.3.1.1	Weighted Precision 100
8.3.2	Results 104
8.4	Need of a Better Relevance 104
8.4.1	An Agent Based Solution 104
8.4.1.1	LawBOT: A Multiagent Assistant for Legal Research 105
8.4.1.2	LawBOT Architecture 106
8.4.2	Using Ontology to Augment Search 109
8.4.2.1	A Proposed Business Ontology 111
8.5	Our Goal 112

Chapter 9

Domain Ontology Construction	118
9.1	Term-relationship Based Extraction 118
9.2	Introduction 119
9.3	Related Work 120
9.4	Our Contribution 122
9.5	Our Approach 122
9.5.1	Related Term-Vector 124
9.5.2	Bayesian Network 125
9.5.3	Ontology Construction 126
9.6	Implementation 126
9.6.1	Weighted Sentence Based Related-Term Vector 127
9.6.2	BN Constructor Algorithm 129
9.6.3	Experiment and Result 131
9.6.4	Using the BN to construct Ontology 132
9.7	Rule-based Extraction 133
9.7.1	Company Ticker Symbol 133

9.7.2	Company Name and Variations	133
9.7.3	Company Product Information	133
9.7.4	Company Executives	135
9.7.5	Synoname	135
9.7.6	Competitors	136
9.8	Conclusion and Future Work	136

Chapter 10

Improving the Keyword Model	138
10.1 Preliminary Improvement	139
10.2 Document-level Relevance	139
10.2.1 Data Set for Document-level Relevance	140
10.2.2 Evaluation	140
10.2.3 Metric	141
10.3 Sentence-level Relevance	141
10.4 Improving Keyword Model Performance	143
10.4.1 Using Ontology	143
10.4.2 Using Variable Window	145
10.5 Conclusion	147

Chapter 11

Interesting Observations	149
11.1 Document Distribution Correlation	150
11.1.1 Document Distribution	150
11.1.2 Correlation	151
11.2 Other Observations	153
11.3 Conclusion	155

Chapter 12

Conclusion and Sentence-based Explanation Model	158
12.1 Why We Need a Separate Model	158
12.2 Sentence-based Model Architecture	160
12.3 Language Model	160
12.4 Explanatory Sentences	163
12.4.1 Language Model	163
12.4.2 Ontology-based Relevance Model	164
12.4.3 Mixture Model	164
12.5 Results	164
12.6 Conclusion	164

List of Figures

2.1	Logarithmic score plot for a World Cup 2002 soccer game between Sweden and Nigeria played on Jun 07	15
2.2	Logarithmic score plot for a World Cup 2002 soccer game between Denmark and France played on Jun 11	16
2.3	Average logarithmic score of fifteen soccer markets on Sports Exchange. Higher (less negative) log scores reflect increasing accuracy. .	17
2.4	Average entropy of fifteen soccer markets on Sports Exchange. Lower entropy values reflect increasing certainty in the outcome.	18
2.5	Average logarithmic score of eighteen basketball markets on Sports Exchange. Higher (less negative) log scores reflect increasing accuracy. .	19
2.6	Average entropy of eighteen basketball markets on Sports Exchange. Lower entropy values reflect increasing certainty in the outcome. . .	20
2.7	Correlation between logarithmic normalized price of the winner and the score difference in the basketball game between San Antonio and LA Lakers held on May 07, 2002	21
2.8	Average logarithmic score of twenty-two political markets on Iowa Electronic Markets. Higher (less negative) scores reflect increasing accuracy. The horizontal axis is the number of days until the end of the market (which occurs on or near election day).	22
3.1	The architecture of the system designed to extract keywords affecting certain events at certain date.	28
4.1	A Web-page from CNN.com and its blocks (shown using boxes) . .	36
4.2	Two Web-pages' block structures as seen by <i>GetBlockSet</i> . The output from them are shown under the dotted line.	45

4.3	Run-times for the <i>LH</i> , <i>ContentExtractor</i> and <i>FeatureExtractor</i> algorithms. The vertical axis represents the time of execution (in seconds) for a number of pages (plotted in the horizontal axis). . . .	59
4.4	Total storage requirement of Web-caches using <i>ContentExtractor</i> -based and Shingling algorithms.	60
4.5	Total execution time taken for <i>ContentExtractor</i> and the Singling-based algorithm.	62
4.6	Accuracy obtained by using our blocking algorithm with an SVM-based classifier based on block features to identify primary content blocks.	63
5.1	A sample Yahoo finance page with temporal expressions highlighted.	71
5.2	Our temporal grammar used in the TimeFinder algorithm.	72
6.1	Analysis procedure for keyword extraction. The date for which we are trying to find an explanation is called the pivotal date. The documents before the pivotal date are in negative document set or Document B(Before) set. The documents after the pivotal date are put in the positive document set or Document A(After) set.	85
7.1	A simple way to show the analysis procedure. The documents are shown as vertical bars of different shades. The documents before the pivotal date (for which date's event we are looking for a reason) are in negative document set or Document B set. The documents after the pivotal date are put in the positive document set or Document A set.	90
7.2	Portion of the price time series for candidate Giuliani in NY Senate election market on IEM, with significant dates highlighted.	91
7.3	Portion of the price time series for candidate Gore in US Presidential election market on IEM. Note that in this market a bet for Gore won if Gore had a larger share of the <i>popular</i> vote.	92
7.4	Portion of the price time series for "100 cents if extra-terrestrial life discovered by 2050" on the Foresight Exchange.	93
8.1	The ranking of an explanatory keyword-set. The top ranked keywords, which can be associated with an explanatory event will be given highest score. The total score is divided by the maximum possible score of 55X (for top 10 keywords) for normalization purpose.	103
8.2	The results from first 10 companies. The score is pretty low for plain vanilla "fixed window" approach.	105

8.3	The architecture of the LawBOT system designed to use ontologies in the context of a legal database search.	107
8.4	A part of the ontology used by the LawBOT system. The ontology here is shown as a connected graph of related keywords.	109
8.5	A proposed ontology for Microsoft	111
9.1	A simple architecture of our system. We create the BN as the knowledge base to be used in the Ontology construction. The main focus of this paper is the part, surrounded by dotted red rectangle.	123
9.2	The BN structure used in our algorithm. Note that to ensure that we construct the DAG we are not connecting w_1 and w_2 even if there is a dependence between them. From w_2 's related term-vector we remove any parent or sibling node before creating the child of w_2	126
9.3	The weighting function $W(j)$ and the set of sentences included in $N(S_i)$ as shown by the bar graphs. The preliminary sentence set was 1, 10, 20, 23, 24, 31, 39, 40, 41, 46, and 50 and the final extended sentence set is 1-3, 10-12, 20-26, 31-33, 39-43, 46-48, and 50. Each bar represents a sentence.	127
9.4	The BN created by our algorithm for eBay. Note that "Paypal", "Amazon", "AspenTech", "Meg Whitman", etc. are all related with "eBay". We show the sub-nodes only for "Amazon" and as we can verify "Sams' Club", "Jeff Bezos", "Yahoo" etc. are also co-related with "Amazon".	131
9.5	A profile page for the company GE (General Electric). We collected useful information from pages like these to build the ontologies of the company.	134
10.1	This figure shows the number of relevant sentences extracted from around 1000 news articles for the company MSFT or Microsoft. We gradually increase the knowledge base capacity from just the ticker symbol to ontology with the use of synonames. As we see the number of relevant sentences extracted increase dramatically by using more and more relevant information.	142
10.2	This figure shows part of the Figure 10.1 blown up to show the effectiveness of using synonames for the company executives.	143
10.3	This figure shows the F-measure for the first 50 news articles from the above chart. The precision and recall values were measured for all the stages. This shows that using ontology and synonyme results in a much higher F-measure than use of just the ticker symbol (which the finance pages of Yahoo usually does).	145

10.4	The improvement in the weighted precision scoring by using the ontology and the variable window approach. The vertical axis represents the “Score” or the “Weighted Precision Score”. The horizontal axis represents the company. FW represents the case for fixed window, FW + Onto shows the case when we apply fixed window approach with ontologies, and VW + Onto is the case where we used variable window approach with ontologies. VW + Onto outperforms the other two cases in almost all companies. . . .	146
11.1	The company names are arranged in ascending order of magnitude of DS or Document/Swing ratio. FW represents fixed window approach. FW+Onto shows the results from fixed window approach with the use of ontologies. VW+Onto uses variable window and ontologies.	152
11.2	Trend comparison of all the keyword models with the DS count value. This graph shows that the VW + Onto model correlates with the DS count most closely. This indicates that the variable width model with the use of ontology can bring more relevant portions of the news articles as the number of news articles increases on average. And more data leads to better analysis and better performance score.	153
11.3	Price graph for HPQ(Hewlett Packard)	154
11.4	Price graph for DELL(Dell)	154
11.5	Price graph for F(Ford Motors)	155
11.6	Price graph for GM(General Motors)	156
12.1	A simple way to show the analysis procedure for explanatory sentence generation. The date for which we are trying to find an explanation is called the pivotal date. Only the documents during or before the pivotal date with same price trend as the pivotal date are considered for the analysis.	161

List of Tables

2.1	Scoring times in Sweden vs. Nigeria and Denmark vs. France games. Time in parentheses is game clock time.	14
2.2	Time difference between the price update and the scoring in 38 different cases of 15 games from 2002 Soccer World Cup resulting in average time difference of 24.7647 seconds. Note that this difference reflects a conservative estimate, due to the thresholding procedure; any delay may be entirely the result of website update delays and/or network delays.	24
2.3	Last Table continued	25
2.4	Correlation between logarithmic score and normalized score difference in 18 basketball games. The average of these correlations is 0.61	25
4.1	Details of the dataset. The number of pages taken from individual categories are not shown due to the enormous size of the latex table, but the interested reader can contact authors to get the details. . .	54
4.2	Block level Precision and Recall values from <i>LH</i> algorithm, <i>ContentExtractor</i> and <i>FeatureExtractor</i> . The second, third, and fourth columns are from <i>LH</i> algorithm, the fifth, sixth, and the seventh columns are from <i>ContentExtractor</i> and the eighth, ninth, and tenth columns are from <i>(K-)FeatureExtractor</i> . We put <i>K</i> in parenthesis to imply that these results are almost same for <i>FeatureExtractor</i> and <i>K-FeatureExtractor</i>	55

4.3	Block level Precision and Recall values from <i>LH</i> algorithm, <i>ContentExtractor</i> and <i>FeatureExtractor</i> . The second, third, and fourth columns are from <i>LH</i> algorithm, the fifth, sixth, and the seventh columns are from <i>ContentExtractor</i> and the eighth, ninth, and tenth columns are from <i>K-FeatureExtractor</i> . Due to poor performance of <i>FeatureExtractor</i> for these Web-pages (which we do not show here) we improved it to <i>K-FeatureExtractor</i>	57
4.4	A property-wise comparison table for three algorithms. Note that <i>(K-)FeatureExtractor</i> here represents both <i>FeatureExtractor</i> and <i>K-FeatureExtractor</i> . For the case of <i>(K-)FeatureExtractor</i> we took the b-precision, b-recall, b-F-measure and all other comparisons with respect to text feature.	58
4.5	A property-wise comparison table for <i>Shingling algorithm</i> and <i>ContentExtractor</i>	61
4.6	Block level Precision and Recall values from <i>Shingling</i> algorithm, and <i>ContentExtractor</i> algorithm for 50 Web-pages. The second, third, and fourth columns are from <i>Shingling</i> algorithm, the fifth, sixth, and the seventh columns are from <i>ContentExtractor</i>	61
5.1	Accuracy values of the output produced by Linear Kernel Support Vector classification. The columns represent Site i.e. the source Web sites where the article was published, Average number of Paragraphs per article , Average number of Sentences per article , Average number of Words per article , Average number of Time Expressions per article , and Accuracy . Here note that sometimes the average number of sentences are really low, but as we randomly check, we realized that sometimes they are long sentences, and sometimes the articles are really short articles as opposed to regular general news articles.	79
7.1	Top features found for dates corresponding to major price changes in the NY Senate market.	95
7.2	Top features found for the US Presidential market.	95
7.3	Top features found for the XLif market on the Foresight Exchange.	95

8.1	Details of the dataset for the randomly selected 20 companies. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model.	106
8.2	Details of the dataset. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model. . . .	113
8.3	Details of the dataset. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model. . . .	114
8.4	Details of the dataset. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model. . . .	115
8.5	Details of the dataset. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model. . . .	116
8.6	Details of the dataset. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model. . . .	117
9.1	Part of the Synoname table used for extracting names abbreviated or used other ways.	136

10.1	Details of the result. Precision from using ontologies are shown in the second column followed by Recall and van-Rijsbergen F-Measure. Precision, Recall and F-Measure using plain vanilla TF-IDF are shown in fifth, sixth, and seventh columns.	141
10.2	Some instances from the keyword model for various companies. As we see keywords are extracted which are very relevant to the facts pertinent to the possible reasons for the price swings for these companies. These results are mainly from variable window approach with the use of ontology (VW + Onto)	144
10.3	Some instances from the keyword model for various companies. As we see keywords are extracted which are very relevant to the facts pertinent to the possible reasons for the price swings for these companies. These results are mainly from variable window approach with the use of ontology (VW + Onto)	147
10.4	Some instances from the keyword model for various companies. As we see keywords are extracted which are very relevant to the facts pertinent to the possible reasons for the price swings for these companies. These results are mainly from variable window approach with the use of ontology (VW + Onto)	148
11.1	Details of the dataset for the randomly selected 20 companies. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model. The last column shows the document distribution versus the number of price up swing or down swing. This is basically the ratio of total file to analyze and the total number of “UP Count” and “DOWN Count”.	151
11.2	One instance from the keyword model for Hewlett Packard is shown here. As Dell and Hewlett Packard are competitors in a flourishing business sector (computer hardware), reports of a bad news of rival (Dell) acted as a boon to Hewlett Packard. Simply, rival’s bad news is good news in a flourishing sector	155

11.3	Two instances from the keyword model for Ford and General Motors, competitors in the auto industry. But the industry is recently not doing well. In this shaky market, if something good happens to any company (here GM), then it influences others as well, even if they are competitors. Here as billionaire Kirk Kerkorian offered to buy a big chunk of General Motors share, not only did General Motors share price go up, but so did it Ford's share. This is a very interesting example of the fact that the market felt more confident in the whole sector as one of the player's (GM's) shares went up due to a good news.	157
12.1	Explanatory sentences for companies for a specified date	163

Acknowledgments

I thank GOD and my parents for everything bestowed upon me. I thank my wife for her love, care and patience. I thank my sisters, brother-in-law for their affection, love and care. I thank my aunty, in-laws, nephew and other family members and friends who have given me hope and cheer. I thank my advisors, teachers, committee members, all my institutions, my country India and the USA for my education and all the opportunities, I have had. I would be nowhere without everybody's help, care, love and support.

Dedication

I dedicate my thesis to my Parents, Wife and my Sisters for their unparalleled love, affection, care, and support.

Chapter 1

Introduction

“The goal of mankind is knowledge ... Now this knowledge is inherent in man. No knowledge comes from outside: it is all inside. What we say a man ‘knows’, should, in strict psychological language, be what he ‘discovers’ or ‘unveils’; what man ‘learns’ is really what he discovers by taking the cover off his own soul, which is a mine of infinite knowledge.”

- Swami Vivekananda

(Hindu Philosopher and Spiritual Leader)
(1863-1902)

After the introduction of the Internet, the amount of online publishing has increased by leaps and bounds. As it is becoming more and more pervasive, the Internet is encompassing all areas of human life. Online publishing has the fascinating inherent feature that content publishing is not controlled by a single entity. As a classic example of a distributed information source, this online publishing is also the largest of its kind on our planet. What this means is that information on the Web is edited, organized and maintained by millions of people with varied backgrounds, styles, organizing skills, and knowledge. As a result, the Web is undoubtedly diverse in domain, class, concept, content and structure. These unique characteristics of distributed control of content and diversity also come with seri-

ous challenges. Searching this huge information source for the right information is just one of these challenges. Traditional search engines crawl the Web, locate documents, store the document or the information in their local cache (most of the time) and create an index of words or phrases present in these documents. When a search request is made with keywords, these engines return these stored documents or links to them (with or without rank, specifying their importance [26, 1, 2, 3, 4]) . However, few of the results returned by a search engine may be valuable to a user [89, 92] who is looking for a deeper understanding of a fact or an event.

Therefore, in this scenario, there are challenges and there are several areas where improvements and advanced features can be introduced. More intelligent crawling [98, 128, 15, 36], parallel crawling [31], focused crawling [54, 28], efficient caching techniques [18], classification of documents [34, 88, 134], useful indexing algorithms [120, 60, 61, 90], categorization of results [20, 25, 97], measuring relevancy of the documents, fast and useful ranking are just a few examples of possible improvements.

While improvements in the aforesaid areas are absolutely necessary and researchers are working on them, the researchers will only improve the result sets coming from a traditional search engine. For example, a focused crawler [54, 28] will only make the crawling more intelligent by making it subject specific, and that can make the search result more specific to the search keyword.

Similarly, an efficient caching algorithm [18] will be useful in storing and retrieving data more economically, efficiently and fast and can provide the user with a copy of the documents, which are removed from their original url location after they are crawled.

A categorization of the result [20] coming from a metasearch engine will improve the readability and accessibility of the documents by situating them properly in a wide variety of categories.

Some of these above improvements are already being tried in commercial search engines. Search engines like Google, Yahoo, and FAST are fetching more and more documents from the Web, by their intelligent crawlers. Thus larger numbers of documents are available to users than before, faster than before. Similarly, search engines like NorthernLight and Vivisimo [5] try to organize their results in an

intelligent and meaningful way to help users focus on certain areas of interest. For example, a search with a phrase like “C Lee Giles” will bring search clusters labelled as “Gary Flake”, “lawrence”, “NEC Research Institute”, “Search Engines”, “David M. Pennock”, “Penn State University” etc. This result will help a user narrow down and focus on a certain specific category, if the user is interested in looking at publications of “C Lee Giles” with “David M. Pennock” or the courses taught by Lee Giles at “Penn State University”.

Though recent commercial search engines are trying to improve the search experience and other features, still they are not enough considering the inherent limitations of the search functionality. After all the improvements, they are still regular keyword-matching search engines, trying to find the best and most appropriate matches for the given keyword(s) among available online documents. We still do not have a full-fledged natural-language question-answering facility available in search engines. There have been recent efforts to provide question-answering services as will be described in the related work section, but a deep analyzing capability of events is still missing in those approaches.

We plan to create a system where users can ask a question similar to: “Why did the Intel stock price fall on Aug 12, 2004?” What this example signifies is that we plan to answer a question of “why” when there is a cause and effect involved regarding the question and the domain involved.

1.1 Motivation

In the present day and age, people are not satisfied with the search results for a keyword. Users find it hard to deal with a search result consisting of several thousand documents. People do not have time to read or analyze them.

Let us assume somebody is looking for IBM stocks and news articles. Not having enough time to read and analyze every news article about IBM, he may want to have an understanding of the whole stock price scenario for the last two years and try to find an explanation for the price movement for a specific date. Can we help him with the existing search technology? In cases like these, even the most advanced search facilities such as those provided by Google [1], Yahoo [2] or other search engines cannot help. These advanced features can only enhance

the search, but cannot analyze the result and produce more intelligent and useful understanding. Therefore, we need a technique or a group of techniques to read, order, analyze, and present an understanding rather than just a keyword-match. To address these issues, we present in this work a group of techniques which can extract the correct right information, order and analyze it and provide a more deeper understanding.

1.2 Problem Statement

We attempt to answer the question “why did a particular event happen?” using information sources, freely available on the Web. As mentioned earlier, by the word “event” we mean an occurrence with which we can associate a time, and so situate it temporally with other events. These events are such that we can also associate a cause and effect relationship. We selected some domains to test the techniques, and attempted to get a deeper understanding of the probable relevant causes and background information for a certain event.

1.2.1 Core Issues

The quality of the Web as a diverse, flexible, open and distributed information source has been seen as a boon to open editing and publishing. The Web also has a very major inherent disadvantage. Most of the Web-text content is written in HTML (or generated as HTML/SGML/XML format from the script) and thus the contents are not tagged intelligently. Even after the introduction and adoption of XML [6], the scenario did not change that much; it will take quite some time before everything is intelligently tagged in a semantically meaningful way. Nevertheless, we do not have a major common standard for content-editing and publishing. The absence of a standard poses a major challenge in Web-data mining.

The Web is very different from a relational database where all the information is stored in tabular, well-structured, and well-defined format. Pulling the required and right information from a database was never a problem (though size, atomicity, consistency, isolation and durability property requirements are important and so intelligent optimization techniques are needed in relational database systems, but

these are separate problems). The simple question of pulling the right information from a Web-page has not yet found a good answer. There have been previous attempts to make complex queries [112] or mapping Web content as a semi-structured database [14, 126] to answer simple questions [127], but they have never been properly scaled [14]. The reason is that answering simple questions from a Web content not only requires mapping the content as a database table [127, 95] but also domain knowledge, understanding the context of the question, language-based techniques and much more. Web search engines are just a very shallow level keyword (or maybe mostly synonym, or related word) matching engine and so left many related questions unanswered. Though there have been attempts to answer natural language queries like “What”, “Who”, “How many”, “Where” etc. ([7] from Zhiping Zheng, AskJeeves [4], IONAUT [8] from Steve Abney and Michael Collins, LAMP [9] from National University of Singapore, QuASM [10] from UMASS/CIIR, Start [11] from MIT, Wondir [12] from Wondir Inc. etc.), a deeper understanding of the domain knowledge based information is missing. That may be the reason why previous researchers never attempted to answer the question “why”.

1.2.2 Contribution

In this thesis, a novel assembly of techniques is proposed that can be applied together on an unstructured information source (in the form of text-based news articles, reports, and other text-based documents). These techniques, applied in a step-by-step fashion, will be able to properly analyze the information source and can provide a text-based explanation for any events occurring during a time-frame. Given a noisy, unstructured, weekly organized source of information, our method can reduce noise with high precision, sort information bits (according to specified attributes) with high accuracy, find relevance with the domain of concern with high precision and in the end provide text based explanations (in the form of keywords, phrases or even with sentences) for a particular event.

1.2.3 Probable Applications

This approach can be used to find a better understanding of the probable causes or factors affecting a particular event in concern. Though we worked mainly with artificial and real market events, the technique can be used in principle as a same set of techniques, as used here, to get a better understanding of the event as well as factors affecting it. The first problem that we worked on is: “The probable causes or factors (in the form of keywords or phrases) behind a certain event in a market”. A rich Web-query system needs more than just finding the keyword or answering simple questions like “What is the GDP of India in 2003?”¹, or “Who is Isaac Newton”, or “What is the distance between New Delhi and New York”. We need an automatic system which can read documents, clean noises in them, earn a better understanding using domain knowledge and give pointers to the factors affecting everyday life.

In the next chapter we will describe the assumptions and demonstrate them with examples from sports betting markets and artificial markets. These assumptions are important to theoretically prove that there is a high correlation between market events and real events and finding the real events affecting the market events will be beneficial in the long run (e.g. we can later come up with a predictive model that can exploit the real events as soon as they are published and act on the market front based on the analysis).

¹Though the most advanced MIT system Start [11] could not answer the question

Chapter 2

Sports Market and Event-Price Correlation

“Research is to see what everybody else has seen, and to think what nobody else has thought”

- **Albert Szent-Gyorgyi**

(Hungarian Biochemist, 1937 Nobel Prize for Medicine)
(1893-1986)

2.1 Prelude

As previously discussed, markets are nice test-beds for analyzing the correlation between an event and price movements. It would be easy just for the experimentation purpose and to prove that the explanations for the events can be justified with the price movements. Prior to the work as described below, no solid proof was described in the literature about the close correlation between the price movements and the real-life events. Real-life events are specified as some of the test-markets used here are not stock-markets but political markets and betting markets. But these markets show very similar and sometimes identical characteristics of a real

market and so the proof of correlation in these markets can prove their existence in real markets too.

2.1.1 Brief Description

This chapter reports on our experiment with these markets. Data is analyzed from thirty-three “interactive” sports betting markets on the World Sports Exchange (WSE), where betting is allowed continuously throughout a sporting event. The study includes markets based on soccer (European football) games from the 2002 World Cup, and markets based on basketball games from the 2002 National Basketball Association (NBA) championship in the United States. Prices in such betting markets are shown on average to approach the correct outcome over time. Important events are recorded throughout the course of the games, for example changes in score. The corresponding price dynamics in the markets are closely coupled with actual game events: the market reacts almost instantaneously to the occurrences of those events, indicating agreement with the assumptions of the efficient markets hypothesis. The dynamics of price changes in soccer games are compared with dynamics in basketball games, highlighting the characteristic differences between these two types of games and their corresponding markets. The nature of these sports betting markets is also compared with “political stock markets” (betting markets on the outcomes of political elections) on the Iowa Electronic Market (IEM) [51, 50]

2.2 Introduction

Typically, a market is thought of as any place where items are bought and sold, whether a physical place or an online one, and whether the items are physical items or more abstract financial instruments like securities or derivatives. At the same time, a market can serve as a tool for aggregating the players’ (i.e., buyers’ and sellers’) knowledge about items of uncertain value. As an example, one of the motivations for a seller to utilize an auction mechanism is to help price an item whose value is not completely known to the seller: rather than set an arbitrary price, the seller can initiate an auction to find out the “correct” (efficient) price

according to the actual demand among the buyers. In fact, under certain assumptions, an equilibrium price in a market can be viewed as a summary statistic that reflects the overall knowledge of all market players about the item's value. So in this sense markets are good mechanisms for combining information that is spread across a population, and summarizing that information in terms of price.

Perhaps an even more direct example of an uncertain-value item than an auction is a *gamble*. A gamble, also called a *security* in the classical economics literature, pays an amount contingent on some future outcome. For example, the gamble “\$1 if it rains tomorrow” pays \$1 if it rains tomorrow, and \$0 if it does not rain. If an agent purchases (one unit of) this gamble for \$0.3, then the agent wins $\$1 - \$0.3 = \$0.7$ if it rains, and loses the \$0.3 otherwise. Clearly the expected value (or expected utility) of the gamble is uncertain, and depends on the buyer's subjective probability of rain.

Tomorrow, the value of the gamble will be certain (exactly \$1 or exactly \$0). But the value of the gamble today depends on the probability of rain tomorrow. In fact, assuming the buyer is risk neutral, the value to the buyer is exactly his or her subjective probability of rain, since the buyer's expected value is $E[\$1 \text{ if RAIN}] = \text{Pr}(\text{RAIN}) \cdot \$1 + \text{Pr}(\text{NO-RAIN}) \cdot \$0 = \$\text{Pr}(\text{RAIN})$.

One of the stronger forms of the so-called *efficient markets hypothesis* (sometimes referred to as the *Hayek hypothesis*, [23] and justified using the *rational expectations* equilibrium concept [64], [86]) states that information is incorporated into market prices virtually instantaneously, as soon as it becomes available to any trader. Informally, the reasoning behind this assumption is that, if some trader has superior information that allows him or her to obtain an expected profit at the current price, then he or she will take advantage of the opportunity by appropriately buying or selling, thereby driving prices toward the correct value (equilibrium price) given the new information.

Gambling markets ranging from horse racing markets [129] to standard sports betting markets like the National Basketball Association (NBA) point spread market [63] to an experimental market in the Euro 2000 soccer championship [121] have been analyzed for signs of economic efficiency and information aggregation. In almost all investigations, assessments coming from the market (in the form of prices or odds encoding probabilities or expectations) appear remarkably accurate

and unbiased.

A new form of sports betting –called *interactive betting*–is now becoming popular on the Internet.¹ In this type of market, bets can be placed or revised at any time throughout the corresponding sporting event, even as, for example, teams score points, penalties are called, or key players are injured. Clearly, as significant events happen “on the field”, the probabilities of the gamble outcomes change, and the expected value of the gamble changes. The continuous nature of the market allows prices to nearly instantly reflect the most recent events in the game, and the most accurate likelihoods of the possible game outcome at any given moment.

We analyze price data from thirty-three such interactive betting markets on the World Sports Exchange (WSE),² including fifteen soccer games from the 2002 World Cup held in Japan and South Korea, and eighteen basketball games from the 2002 National Basketball Association (NBA) Championship held in several different cities in the United States. Prices nearly always react almost immediately to game events (e.g., score changes), as predicted by the efficient markets hypothesis. Prices in the markets also inexorably converge (on average) toward the correct answer, as information pertaining to the final outcome is revealed during the course of the game. We highlight distinctive properties of soccer and basketball games and the characteristic differences between their corresponding betting markets, with some implications for the relative psychologies of the two games.

2.3 Background

2.3.1 Related work

It is clear that markets often react quickly to the release of new and relevant information. This becomes very interesting in the case of betting markets as the relationship between price and information is very clear. There is a direct relationship between the current price of a gamble and the probabilities of the possible payoffs of it, and so any information available about the gamble for any possible outcome should affect the price appropriately according to the rules of Bayesian

¹Some say interactive betting will do for gambling what Ebay did for auctions.

²<http://www.wsex.com>

updating.

The economic theory of *rational expectations* (RE) accounts for information incorporation in markets. RE theory posits that prices reflect the sum total of all information available to all market participants [64, 86]. Even when some agents have exclusive access to inside information, prices equilibrate exactly as if everyone had access to all information. The procedural explanation is that prices reveal to the ignorant agents any initially private information; that is, agents learn by observing prices.

Plott et al. [108] investigate, in a laboratory setting, whether parimutuel markets (the type employed at horse races) are able to aggregate information, as postulated by RE theory. Plott and Sunder [109, 110] and Forsythe and Lundholm [58] conducted laboratory experiments to test the reasonableness of the RE assumption in the context of a securities market (essentially a betting market as described in the introduction). In many cases in all these experiments, the equilibrium reached reflected the combination of all information, as predicted by RE theory.

Beyond the controlled setting of the laboratory, empiricists have analyzed the accuracy of implied probability assessments given by public markets. Perhaps the most direct tests involve betting markets. Several studies demonstrate that odds on horses at the track correlate well with the actual frequencies of victory [129]. Other sports betting markets, like the National Basketball Association point spread market [63], or an experimental soccer market organized at the Max Plank Institute in Germany [121], provide accurate and unbiased forecasts of likely game outcomes. Financial options markets (in many ways equivalent to betting markets) yield accurate probability distributions over the future prices of their underlying stocks [76].

The Iowa Electronic Market (IEM) supports trading in securities tied to the outcome of political and financial events. Since opening to the public, many IEM election markets (especially US Presidential election markets) have attracted wide participation and following, and in many cases have proven more prescient than public opinion polls [59]. Other election markets have now opened in Canada³ and

³<http://esm.ubc.ca/>

Austria.⁴ Pennock et al. [102] show that forecasts on IEM consistently improve over time, reflecting a roughly constant flow of information into the market. The authors develop a theory of information aggregation to explain this behaviour, and describe an algorithm to automatically extract semantic explanations of large market swings by mining online news sources.

Even market games, run entirely with play money, show signs of “economic” efficiency and information aggregation. Pennock et al. [104, 105] show that games like the Hollywood Stock Exchange⁵ and the Foresight Exchange⁶ (an implementation of Hanson’s idea futures concept [66]) yield remarkably accurate forecasts of future events.

2.3.2 Metrics

The current work uses the *logarithmic score* to measure accuracy and information incorporation in betting markets. The logarithmic score is a *proper scoring rule* [132], and is an accepted method of evaluating probability assessments. When experts are rewarded according to a proper score, they can maximize their expected return by reporting their probabilities truthfully. Additionally, more accurate experts can expect to earn a higher average score than less competent experts. Suppose an expert reports probabilities $p_1, p_2, \dots, p_k$ for k mutually exclusive and exhaustive alternatives. Let $w_i = 1$ if and only if the i th event occurs, and $w_i = 0$ otherwise. Then the expert’s score for the current event is

$$\ln \left(\sum_{i=1}^k w_i p_i \right). \quad (2.1)$$

Higher scores indicate more accurate forecasts, with 0 the maximum and negative infinity the minimum. The “expert assessments” given by the market are taken to be the (normalized) prices of the possible outcomes.

Note that under the logarithmic scoring rule, an expert’s expected score equals the entropy of his or her probability distribution. Stated another way, the negative of the logarithmic score gives the amount that the expert is “surprised” by the

⁴<http://ebweb.tuwien.ac.at/apsm/>

⁵<http://hsx.com/>

⁶<http://ideosphere.com/>

actual outcome. So the logarithmic score is both a measure of forecast accuracy and an information-theoretic measure of the amount that the market is surprised when the winner of the gamble is finally determined.

We used the midpoint of the bid and ask prices of the eventual winner of the bet to calculate the average logarithmic score. If the price (bid+ask/2) of the ultimate winner is $p(t)$ at time t , then the average logarithmic score for the market at time t is

$$\text{Average_Logarithmic_Score}(t) = \frac{\sum_{i=1}^N \log p(t)}{N} \quad (2.2)$$

where N is the number of markets. Similarly, the average entropy at time t is

$$\text{Average_Entropy}(t) = \frac{\sum_{i=1}^N -p(t) \log p(t) - (1 - p(t)) \log(1 - p(t))}{N} \quad (2.3)$$

2.4 Soccer World Cup 2002

Fifteen markets on the WSE were analyzed corresponding to fifteen soccer games in the 2002 World Cup. These include several games from the first round and a few from second round of the tournament. The markets for the soccer games on average started long before the games started (usually 1-12 hours before the commencement of the game). The markets offered continuous betting on the outcome of the game, often with an associated point spread. A game of soccer normally lasts for 90 minutes in the first round (where games can end in a win for either team, or a draw). In the later rounds of the tournament a game may continue for more than 90 minutes if it remains undecided. In that case, the game will be played for 30 minutes more; if it still remains undecided, then the decision will be based on a penalty shoot-out. In the games we considered here, there are no cases of a penalty shoot-out. The matches were played from June 7 to June 15, 2002, and three out of these fifteen games ended in draws. Price variations from the WSE market, the score changes in the actual game, and the game clock time were recorded in parallel. The prices, scores, and game clock are aligned according to the time at which the information was gathered from the World Wide Web. For scores and game clock information, Livescore.com was used. Prices, scores, and clock information were gathered in ten second intervals throughout the games, though

Date	Match Details	Scoring	Time (minutes)
June 7, 2002	Sweden vs. Nigeria	0-1	28
	Sweden vs. Nigeria	1-1	36
	Sweden vs. Nigeria	2-1	79
June 11, 2002	Denmark vs. France	1-0	22
	Denmark vs. France	2-0	85

Table 2.1. Scoring times in Sweden vs. Nigeria and Denmark vs. France games. Time in parentheses is game clock time.

the websites did not always update immediately, and network delays may have introduced further synchronization errors.

Examples of price changes for two representative games are shown in Figures 2.1 and 2.2. Figure 2.1 is the logarithm of the price of Sweden (the bet winner) in the “Sweden vs. Nigeria” game played on June 7, 2002, while Figure 2.2 is the same for Denmark in the “Denmark vs. France” game played on July 11, 2002. Interestingly, in the first game (“Sweden vs. Nigeria”) there are immediate and large price variations after each scoring, but in the second game (“Denmark vs. France”), though price changes are prominent for the first goal, there is almost no price change after the second goal. This highlights the natural time factor involved in the market where the team currently winning near the end of the game is likely to finish as the winner, and, depending on the score, a last-minute goal does not have much impact on the outcome. The scoring times in these two games are listed in Table 2.1. For scoring in the second half of the game, both the actual time from the start of the game and the game clock time are shown in parentheses. Other games display very similar characteristics to these two games.

2.4.1 Delay Calculation

This work evaluates how promptly the market reacts to scoring in the games. The timestamp of the scoring is recorded as τ_s and the timestamp of the price update as τ_p . The network delay supposedly associated with these two websites is represented as δ_s and δ_p . It is assumed that no general predicting devices have been used while playing in the market and any negative time difference between τ_p and τ_s is due to the delay difference in updating of the corresponding websites. This difference is eliminated by thresholding the most negative different at 0, and adding this value

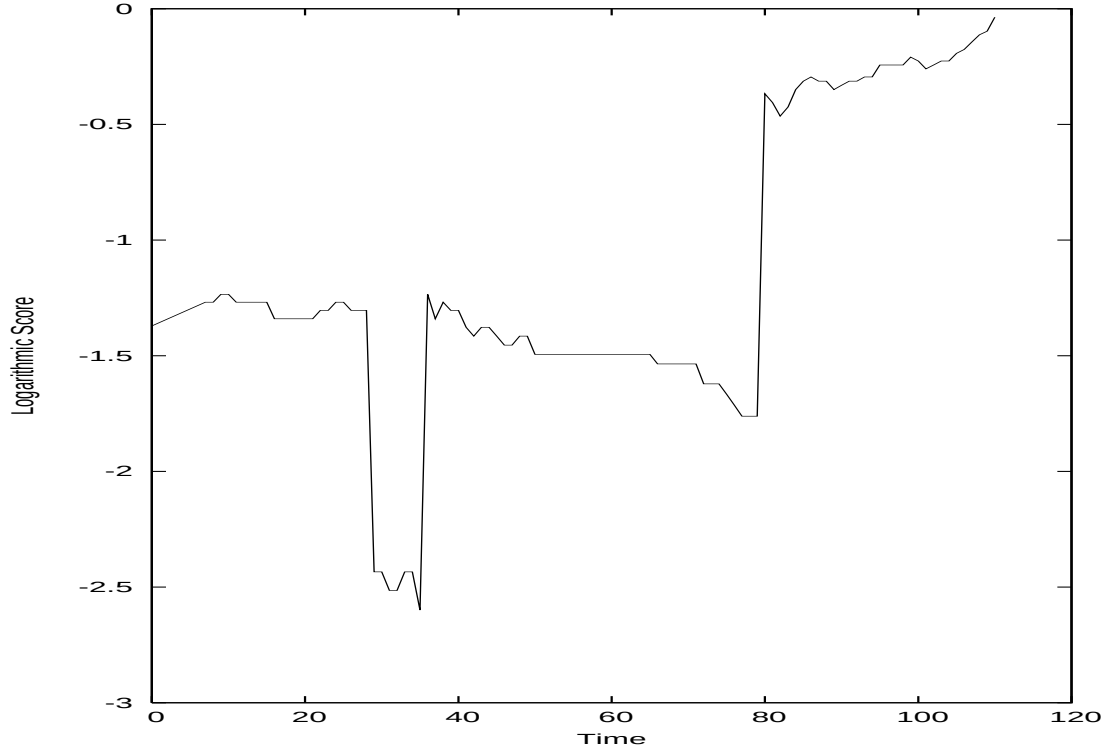


Figure 2.1. Logarithmic score plot for a World Cup 2002 soccer game between Sweden and Nigeria played on Jun 07

to every other difference; this results in a conservative (over-) estimate of delay. Thus the delay Δ is defined as

$$\Delta = (\tau_p - \tau_s) + \Theta \quad (2.4)$$

where $\Theta = \max(\tau_s - \tau_p)$ is the threshold. All 38 cases of these are shown in Tables 2.2 and 2.3. Column 2 shows the date when the game was played, column 3 shows the details (team names), and column 4 shows the scoring; in case of multiple goals, it shows them one after another with the final score last. Columns 5 and 6 give the threshold and delay in each individual cases. It was found that the average delay is 24.7647 seconds. Note that this difference reflects a conservative estimate, due to the thresholding procedure; any delay may be entirely the result of website update delays and/or network delays. Note also that in the calculation cases where there was no change of price after goal scoring were excluded (i.e.,

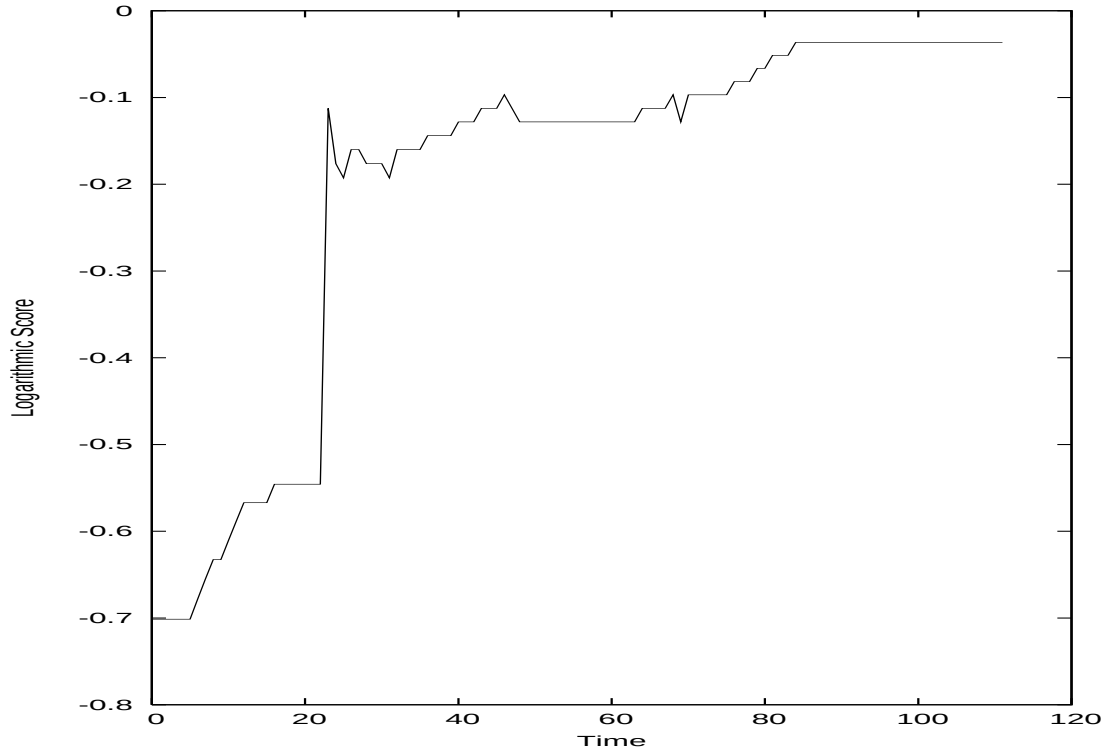


Figure 2.2. Logarithmic score plot for a World Cup 2002 soccer game between Denmark and France played on Jun 11

when the outcome is already nearly certain).

The average log score and the average entropy of all 15 soccer games appear in Figures 2.3 and 2.4. Both are plotted versus time, where 0 is aligned with the start of the game. Price movement information during the interval, which typically consists of two halves of 45 minutes of play separated by a 15 minute halftime break was processed. Surprisingly, some minor price movements were found after the 90 minutes of play ended, so the plots range in time (in minutes) from 0 to close to 110. As seen from the graph, it is clear that during the time period from the 46th to the 61st minute (which approximately encompasses halftime of most games) there is less price movement (the average log score and entropy are flat compared to other regions), as expected, since very little information about the game outcome is decided during halftime. Moreover, the final few minutes show large price movements, meaning that the largest changes in the amount of certainty of the outcome happens at the end of the games on average (around

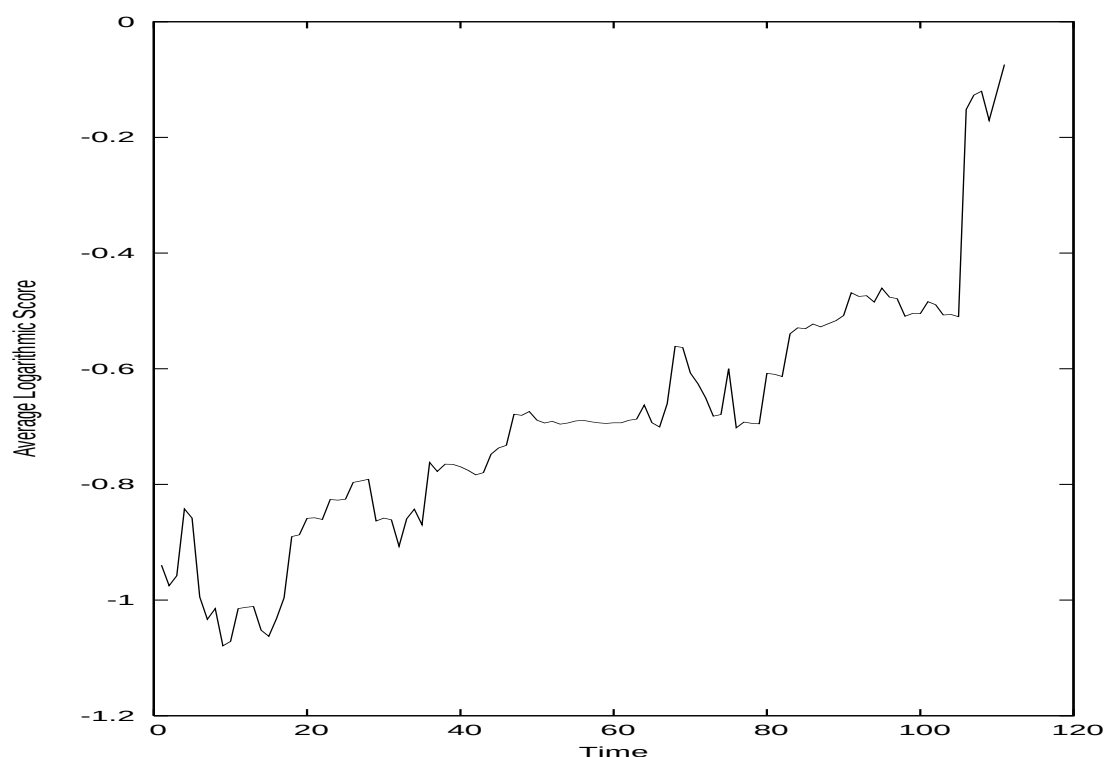


Figure 2.3. Average logarithmic score of fifteen soccer markets on Sports Exchange. Higher (less negative) log scores reflect increasing accuracy.

104th minute in the graph which actually is (104 - 15) or 89th minute of the 90 minute game). This indicates a quite clear and intuitive fact that people feel more certain about the outcome of the game just a few minutes before the game ends. Otherwise the average log score is increasing roughly linearly heading toward zero, indicating a roughly constant flow of information on average through much of the game. The average entropy of these markets (Figure 2.4) shows the downward slope during the two halves of play as information about the outcome becomes available, and the ultimate drop to 0 entropy (certainty) at the end of the game. This indicates a roughly constant reduction of uncertainty over time with a more rapid resolution of uncertainty near the end of the game. The small glitch near the end is due to instances where last minute goals changed the result of the game and the uncertainty level increased slightly for a few minutes.

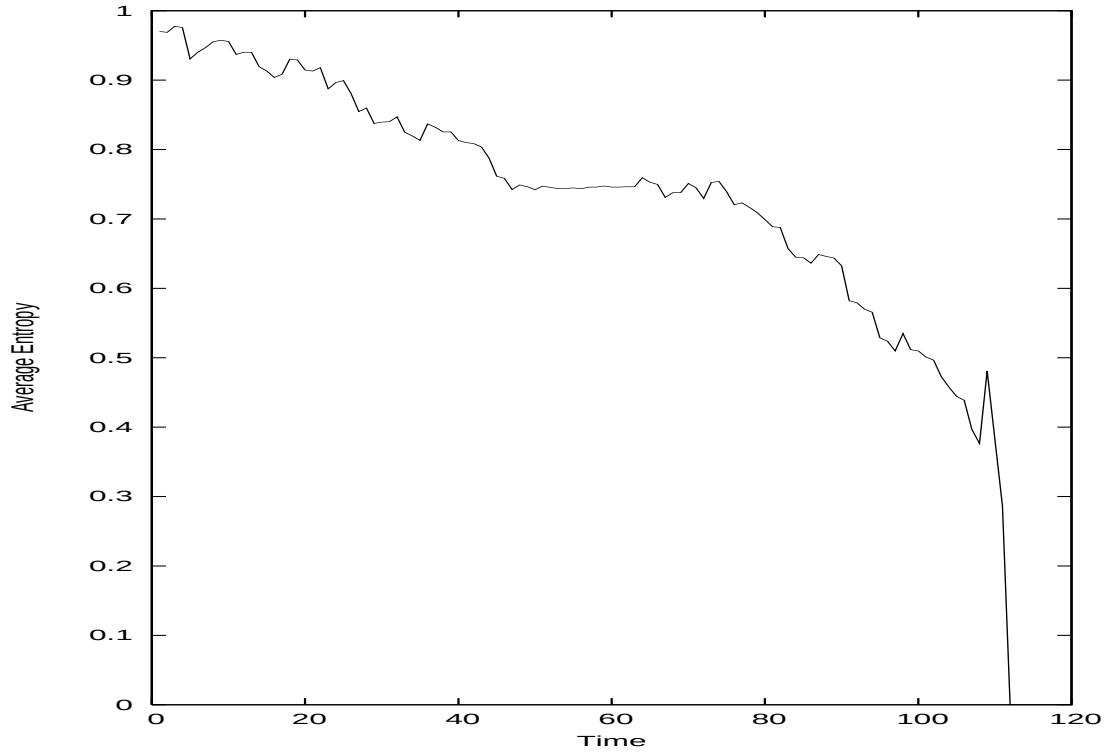


Figure 2.4. Average entropy of fifteen soccer markets on Sports Exchange. Lower entropy values reflect increasing certainty in the outcome.

2.5 2002 NBA Championship

Basketball games are characteristically different and played for 4 quarters, each lasting for 12 minutes, totalling 48 minutes of playtime. There is a halftime break of about 20 minutes in between the 2nd and 3rd quarters. We analyzed 18 games from the 2002 NBA championship played between May 6, 2002 and May 31, 2002. Similar to the soccer games, the score and game clock information was taken from Sportsline.com and the price information from WSE, and they were aligned according to the timestamps. On average the stream was sampled every 10 seconds. Figure 2.5 plots the average logarithmic score and Figure 2.6 plots the average entropy of these games, using the average of the bid and ask prices $(\text{bid} + \text{ask}/2)$. Here time is shown in minutes starting from 0 to 170 minutes. Though the actual game is played for only 48 minutes, here price information was also processed during several breaks, which adds up to a little more than 2.5 hours in total play

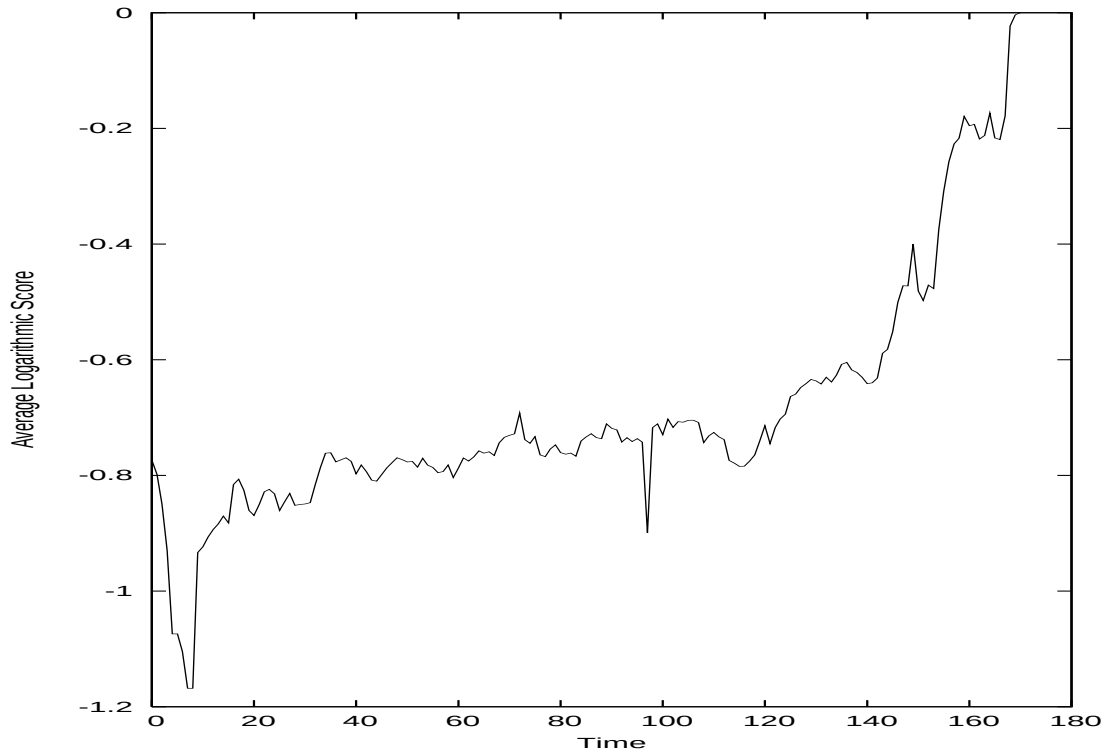


Figure 2.5. Average logarithmic score of eighteen basketball markets on Sports Exchange. Higher (less negative) log scores reflect increasing accuracy.

time.

The average entropy of these games (Figure 2.6) is indicative of major uncertainty over the course of the game as the entropy value is greater than 0.7 for more than 77% of the game, and greater than 0.8 for more than 55.5% of the game. This means the people participating in the market are very uncertain about the outcome of the game and their uncertainty is reflected in the price movements. Not surprisingly this entropy drops sharply over the last 23% of the game and heads towards 0, reflecting increased certainty about the outcome in last quarter of play.

Results show that price changes in basketball games are very much correlated with score changes. Figure 2.7 displays the correlation in one of the games on May 7, 2002, played between San Antonio and the Los Angeles Lakers. This is one of the 18 games where the correlation between the price on WSE and the score differential in the game is very high. In this graph the top curve is the normalized

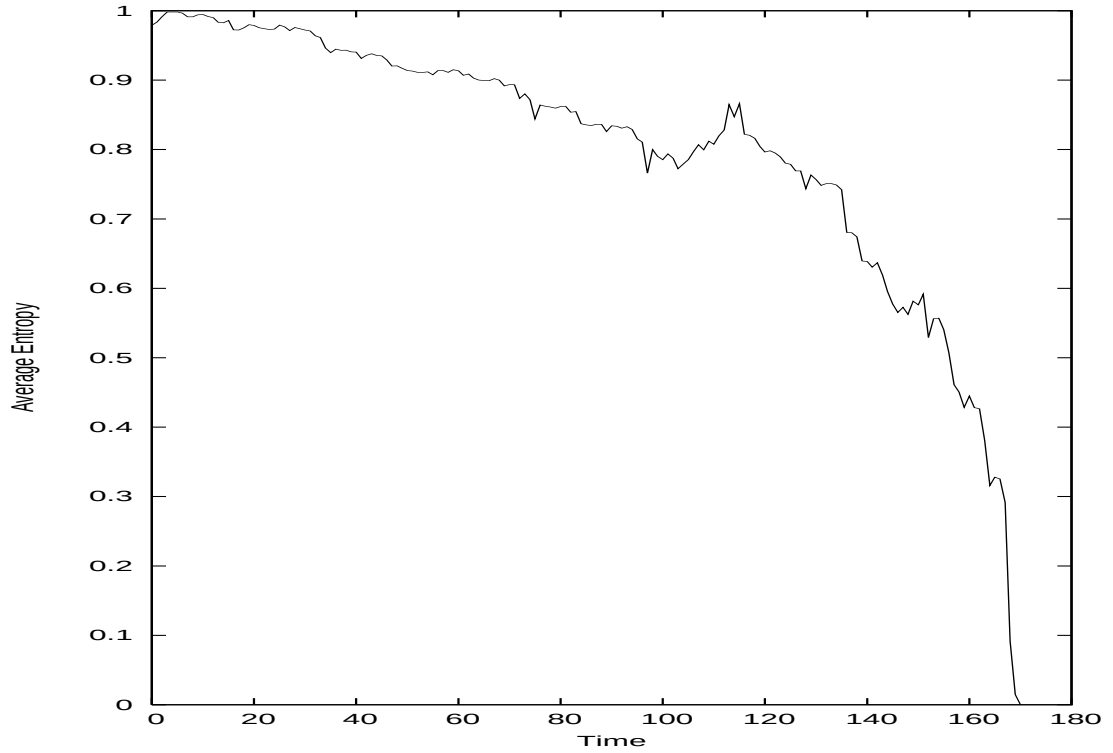


Figure 2.6. Average entropy of eighteen basketball markets on Sports Exchange. Lower entropy values reflect increasing certainty in the outcome.

score difference between the two teams and the bottom curve is the logarithm of the price of the winner. The correlation between these two has been measured as 0.93. For other games the correlation values are shown in Table 2.4.

2.6 Soccer and Basketball

We will now try to compare the basic characteristics of soccer and basketball. In the last two sections the average entropy and average logarithmic scores of these two types of markets were illustrated. The correlation between scoring and price movements has also been shown. Following are some observations about the similarities and differences between the two types of games.

- Basketball games are more uncertain for a larger proportion of the game. One way to interpret this is that basketball is more exciting longer into the

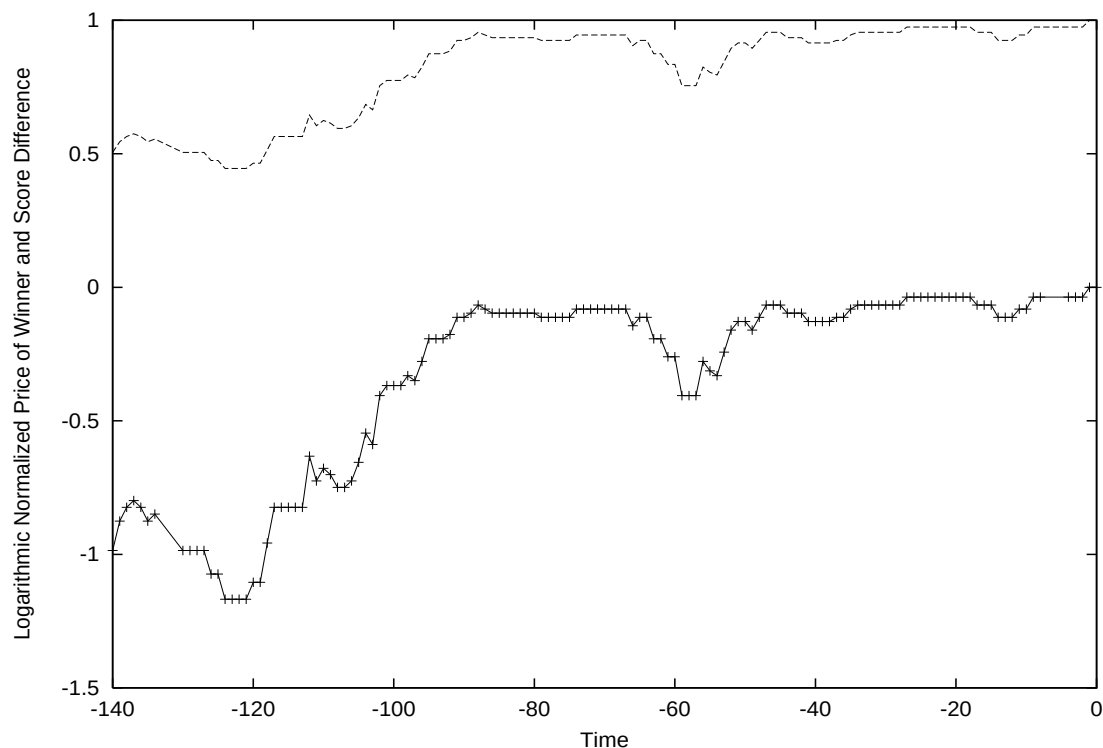


Figure 2.7. Correlation between logarithmic normalized price of the winner and the score difference in the basketball game between San Antonio and LA Lakers held on May 07, 2002

contest, as the outcome does not become clear until late in the game. Putting it other way, when a comeback does occur in soccer, it is that much more unlikely, and thus more dramatic than in basketball.

- Both in basketball and soccer the price is highly correlated with the scoring, but there are some fundamental differences between these two due to the fundamental difference of these two games. In the case of basketball, there is continuous scoring throughout the game and so the price change is very frequent, following closely with the scoring. Typically in basketball a single score for either team does not change the price much. In the case of soccer markets, prices do not change that often, but when they do change with a score it is often a very drastic change.

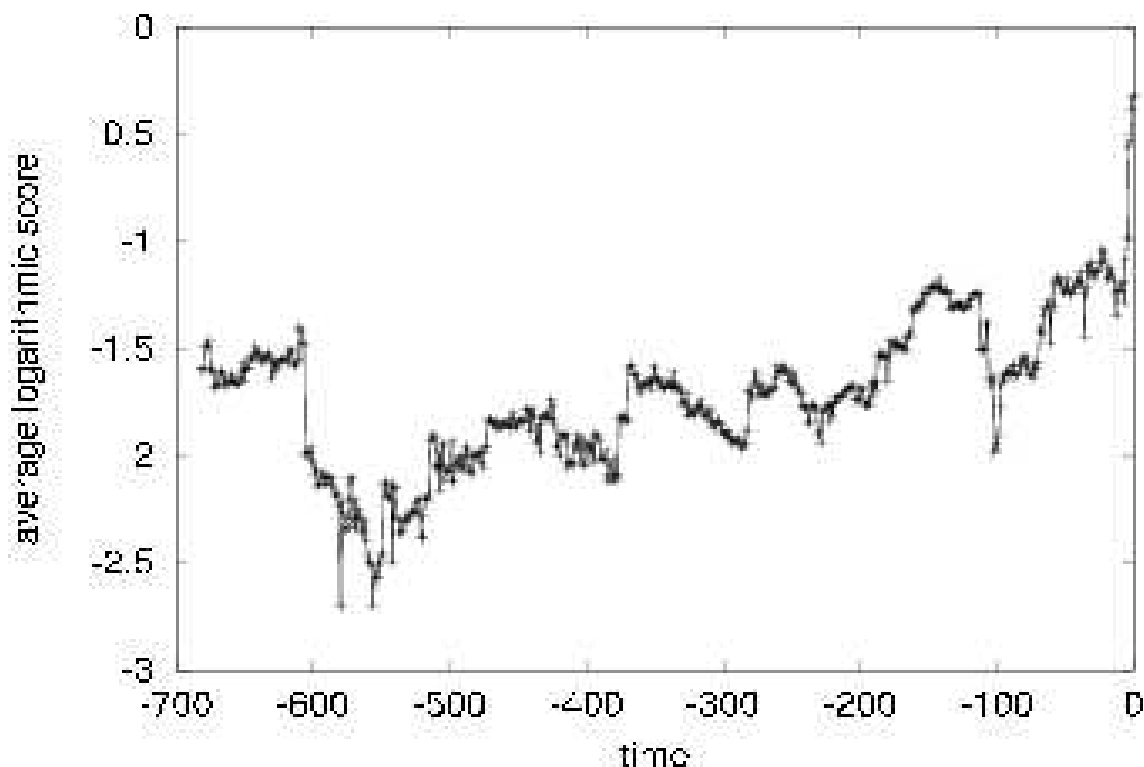


Figure 2.8. Average logarithmic score of twenty-two political markets on Iowa Electronic Markets. Higher (less negative) scores reflect increasing accuracy. The horizontal axis is the number of days until the end of the market (which occurs on or near election day).

2.7 WSE and the Iowa Electronic Market

The Iowa Electronic Market (IEM) is a betting market mainly focused on political elections, run for research purposes by the University of Iowa Tippie College of Business.

Figure 2.8 shows the average logarithmic score of twenty-two IEM markets. There is a qualitative similarity between this graph and the average logarithmic score of the markets on the WSE. One difference between these markets is that IEM markets are longer term (run over a period of months or years), while the sports markets are shorter term (run over a period of hours). The horizontal axis of the IEM markets measures the number of days; that of the WSE markets, measured in minutes. Nonetheless, we still see a roughly constant increase in log score on average with a large increase near the end as election day approaches

and the outcome becomes clear. In previous work [102], an entropy-based feature extraction algorithm was employed to mine possible semantic explanations from news sources for large price changes in the 2000 US New York Senate market and 2000 US Presidential market.

2.8 Conclusion

This chapter has set out data analysis from sports betting markets, where betting is allowed continuously throughout the sporting event. The analysis has covered both soccer games and basketball games. It has been shown that prices in such betting markets on average approach the correct outcomes. Important events throughout the course of the game (goal scoring, etc.) and the corresponding price dynamics are well-correlated, and the market reacts almost instantaneously to these events. It has also been shown that the distribution models of soccer and basketball exhibit some characteristic differences. Still, many characteristics are qualitatively similar, including in much longer-term markets like the IEM.

The easy part of the sports betting market is that the real events which are responsible for the price dynamics are basically the scores of the ongoing games. Discovering the correlation between price and the game scores (real life events), or viewing from another angle, the explanation of the price change from the game scores or real life events, is easier. Here the numbers tell the truth. But it is much harder if the real life events can not be quantified by an integer number, as in a game scorecard. The next chapter looks into this much harder version of the problem. For example in real stock exchanges or even other markets such as a political market, or a futures market, where real life events affecting the price are related news, the problem becomes multiplyingly harder. As one cannot quantify the effectiveness or score of any event against another as easily as above, other efforts need to be made to come close to that goal.

No.	Date	Match Details	Score	Threshold Θ	Time Difference Δ
01	June 7, 2002	Argentina vs. England	0-1	56 sec	0 sec
02	June 7, 2002	Sweden vs. Nigeria	0-1	11 sec	55 sec
03		Sweden vs. Nigeria	1-1	11 sec	2 sec
04		Sweden vs. Nigeria	2-1	11 sec	0 sec
03	June 8, 2002	Brazil vs. China	1-0	0 sec	0 sec
06		Brazil vs. China	2-0	0 sec	22 sec
07		Brazil vs. China	3-0	0 sec	0 sec
08		Brazil vs. China	4-0	0 sec	no change
09	June 8, 2002	Italy vs. Croatia	1-0	44 sec	44 sec
10		Italy vs. Croatia	1-1	44 sec	0 sec
11		Italy vs. Croatia	1-2	44 sec	no change
12	June 8, 2002	S.Africa vs. Slovenia	1-0	0 sec	0 sec
13	June 9, 2002	Costa Rica vs. Turkey	0-1	44 sec	0 sec
14		Costa Rica vs. Turkey	1-1	44 sec	22 sec
15	June 9, 2002	Japan vs. Russia	1-0	34 sec	0 sec
16	June 10, 2002	Korea vs. USA	0-1	0 sec	21 sec
17		Korea vs. USA	1-1	0 sec	0 sec
18	June 10, 2002	Portugal vs. Poland	1-0	0 sec	22 sec
19		Portugal vs. Poland	2-0	0 sec	0 sec
20		Portugal vs. Poland	3-0	0 sec	0 sec
21		Portugal vs. Poland	4-0	0 sec	no change
22	June 10, 2002	Tunisia vs. Belgium	0-1	0 sec	21 sec
23		Tunisia vs. Belgium	1-1	0 sec	0 sec
24	June 11, 2002	Denmark vs. France	1-0	0 sec	0 sec
25		Denmark vs. France	2-0	0 sec	no change
26	June 11, 2002	Senegal vs. Uruguay	1-0	41 sec	0 sec
27		Senegal vs. Uruguay	2-0	41 sec	83 sec
28		Senegal vs. Uruguay	3-0	41 sec	124 sec
29		Senegal vs. Uruguay	3-1	41 sec	83 sec
30		Senegal vs. Uruguay	3-2	41 sec	83 sec
31		Senegal vs. Uruguay	3-3	41 sec	62 sec

Table 2.2. Time difference between the price update and the scoring in 38 different cases of 15 games from 2002 Soccer World Cup resulting in average time difference of 24.7647 seconds. Note that this difference reflects a conservative estimate, due to the thresholding procedure; any delay may be entirely the result of website update delays and/or network delays.

No.	Date	Match Details	Score	Threshold Θ	Time Difference Δ
32	June 12, 2002	S.Africa vs. Spain	0-1	11 sec	33 sec
33		S.Africa vs. Spain	1-1	11 sec	35 sec
34		S.Africa vs. Spain	1-2	11 sec	0 sec
35		S.Africa vs. Spain	2-2	11 sec	11 sec
36		S.Africa vs. Spain	2-3	11 sec	33 sec
37	June 13, 2002	Ecuador vs. Croatia	1-0	0 sec	86 sec
38	June 15, 2002	Germany vs. Paraguay	1-0	11 sec	0 sec

Table 2.3. Last Table continued

No.	Date	Winner	Details	Correlation
01	May 6, 2002	Dallas	Dallas vs. Sacramento	0.911234895882537
02	May 7, 2002	New Jersey	Charlotte vs. New Jersey	0.931731122180932
03	May 7, 2002	San Antonio	San Antonio vs. LA Lakers	0.643892830218919
04	May 10, 2002	Detroit	Detroit vs. Boston	0.241489294093237
05	May 10, 2002	Los Angeles	Los Angeles vs. San Antonio	0.899529974086272
06	May 12, 2002	LA Lakers	LA Lakers vs. San Antonio	0.769319276995048
07	May 13, 2002	Sacramento	Dallas vs. Sacramento	0.913128573373542
08	May 14, 2002	Boston	Boston vs. Detroit	0.759604460761744
09	May 14, 2002	San Antonio	San Antonio vs. LA Lakers	0.366704064261902
10	May 15, 2002	New Jersey	Charlotte vs. New Jersey	0.764031988154954
11	May 24, 2002	Sacramento	Sacramento vs. LA Lakers	0.794065198465937
12	May 25, 2002	New Jersey	New Jersey vs. Boston	0.469322845099931
13	May 26, 2002	Sacramento	Sacramento vs. LA Lakers	0.15290543696519
14	May 27, 2002	New Jersey	New Jersey vs. Boston	-0.11302313355228
15	May 28, 2002	LA Lakers	LA Lakers vs. Sacramento	0.680983360751506
16	May 29, 2002	New Jersey	Boston vs. New Jersey	0.851181768248148
17	May 31, 2002	New Jersey	New Jersey vs. Boston	0.876953977582046
18	May 31, 2002	Sacramento	Sacramento vs. LA Lakers	0.193926069160131

Table 2.4. Correlation between logarithmic score and normalized score difference in 18 basketball games. The average of these correlations is 0.61

Chapter 3

System Design

“Design is what you do when you don’t [yet] know what you are doing.”
- **George Stiny**
(Professor of Architecture, Massachusetts Institute of Technology)
(June 21, 2002)

As will be seen in the coming chapters, finding real life events in textual format to explain price movements is a much harder and complex version of a simple sports betting market and game score. The reasons behind this complexity are many. Firstly, unlikely in a sports event, which is the scorecard of the ongoing game, finding relevant and correct information is the main problem. The second major obstacle is to align the price movement and the events properly. An efficient market is such that any new information will be immediately incorporated into the market price, and so finding the “right information” at the “right time” is most crucial in this problem. Once the right information is aligned with the right timeline, the information can be analyzed, compared with other similar events, and the patterns and/or the information content¹ in them can be discovered to justify the reasoning behind any price dynamics. So the main thrust of this research is to find

¹Information content is basically the textual content in the news articles

the right information aligned with the right time-line.

3.1 Description of the Architecture

In Figure 3.1, the basic overall architecture of the system is shown. Each block in this architecture represents each subtask of the whole system. The whole system depends on these blocks or subtasks as each of these blocks can potentially improve overall system performance. Therefore, to achieve the efficiency and high accuracy of the whole explanation model, it is necessary to concentrate on each of these blocks and try to improve these subtasks' efficiencies. In the coming chapters, we will take each individual subtask of this architecture, explain in detail why we need the subtask and how we improved their performance. Here we will briefly mention their function, necessity, and their intended use.

3.1.1 Getting the Information

We used basic web-crawler mechanisms [31] to download web-documents that are mainly news articles for events. The two most important facts to mention here are (1) First, the whole approach depends on the news articles. Sometimes we could not find enough articles due to various reasons ². The efficiency of our approach suffered in few places. (2) Secondly, though these news articles are the most important source for up-to-date knowledge, yet they are unstructured in nature and so filtering the noise out and getting the right information is one of the most important tasks in our approach.

3.1.2 Getting the Right Information: Filtering Out the Noise

The first problem and subtask in this architecture is to filter in the proper information about the event and filter out the noise associated with it. Online news articles are the main source of “event-data”. The positive side of this is that they are a rich source of information and are freely available. The negative side is the

²crawling restrictions, subscription requirements, broken links, very few filtered documents, etc.

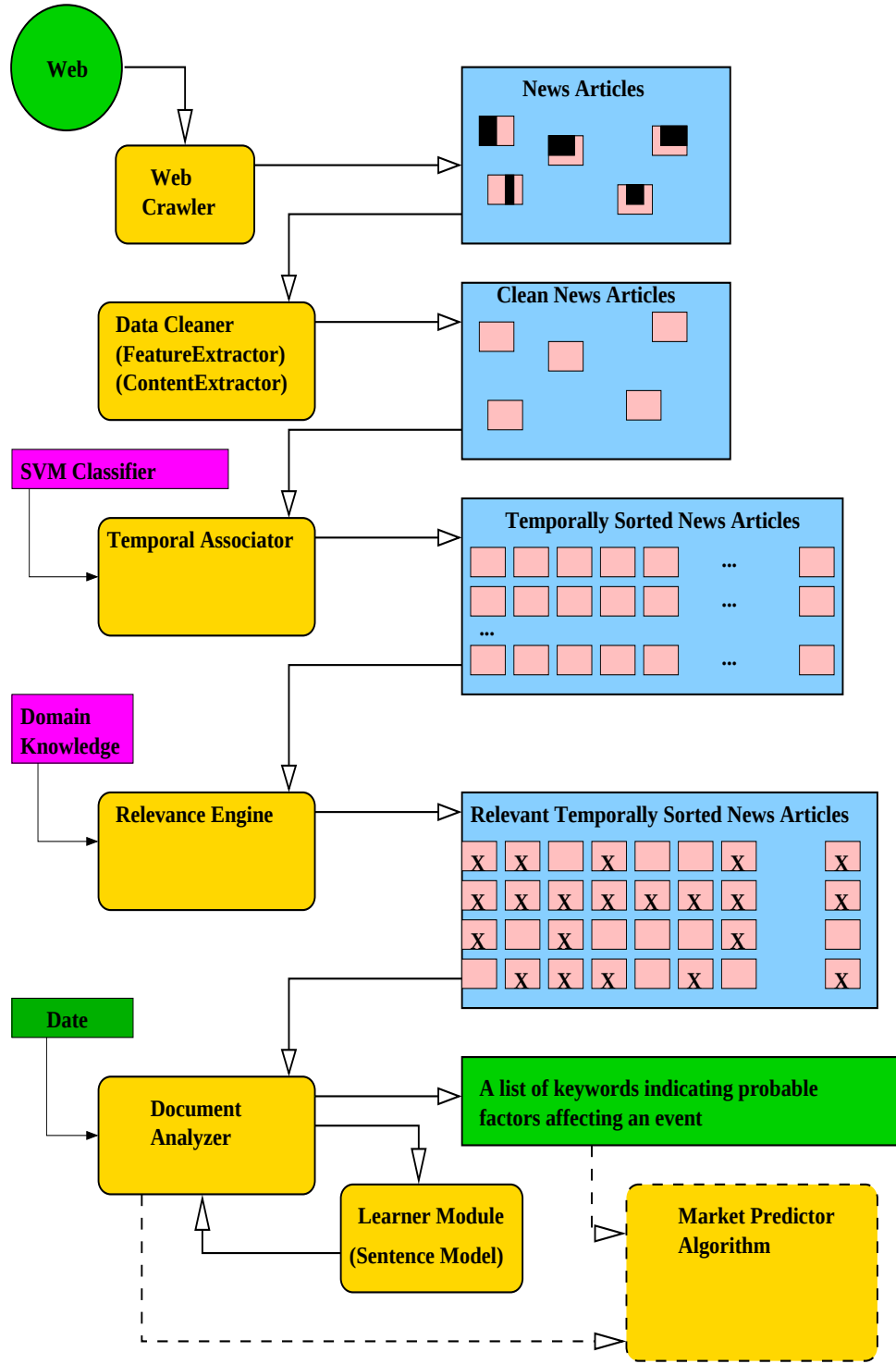


Figure 3.1. The architecture of the system designed to extract keywords affecting certain events at certain date.

amount of noise in the form of banner-advertisements, images, javascripts, applets and various other HTML elements. So in this subtask, the very first step is to remove the noise so that we can access the right information. By crawling the online news articles from various sources, one finds that they are naturally full of irrelevant textual or non-textual blocks of data. These constitute a huge portion of the data ³. Here in Figure 3.1, this is shown as the “Filter-block”. This step of the process is described in the next chapter 4.

In the next chapter we will lay out a plan, devise the algorithm for text-based information extraction in detail, and prove our method’s high performance with crawled data. This algorithm can improve the performance of the overall system by reducing noise content (banner-advertisements, header, footer and any other redundant information) by cleaning the textual data of these redundant contents.

3.1.3 Right Information at the Right Time: Temporal Ordering

Just getting the right information is not enough in this system. We need to align the articles with the proper temporal information. The question of “Why did it happen” can only be answered if we know *exactly* “When did it happen”. The right information is useless when temporal information is of vital importance (which is always true in most time-related cause-and-effect events). As price dynamics are precisely time-dependent, we need to align the real-life events in that order. Thus extracting temporal information and ordering news articles by their date and time is one of the most crucial parts of the system. This temporal ordering process is represented in the architecture as the “Temporal Ordering” block. We will elaborate more on this in chapter 5.

3.1.4 Getting the Relevant Information: Relevance Measure

Getting the right information also means getting the right information for the right candidate. Simply put, in the sports betting market, the scorecard of **China vs.**

³If we just compare the sizes of an HTML page before and after the cleaning we see an 86% size reduction (See chapter 4)

India is of no use when analyzing the market of **USA and Mexico**. Similarly, in analyzing data from the real stock market, if one is analyzing the news data of “IBM”, then a news article on “BBBY” (Bed, Bath and Beyond) is not relevant. Things are complex in the stock market and sometimes people are influenced by different news. But it is the right amount of domain knowledge which will be crucial in deciding whether a news story is relevant or irrelevant. We will first attempt the problem with regular keyword-dependent model without any domain knowledge. Later we will introduce domain knowledge and will show how introducing it will improve performance. The idea of introducing domain knowledge is also an improvement over the existing TDT (Topic Detection and Tracking) research regarding sentence based relevance. We will show how domain knowledge in the form of ontology [52] can be incorporated and can play a crucial role in improving the relevance measure. Therefore, using domain knowledge, the news articles can be narrowed down in more detail, and put in proper bins⁴ of the companies where they should belong.

3.1.5 Building Domain Ontology

Building domain ontologies is a necessary subtask of our approach as will be shown in the coming few chapters. Building an ontology could be a enormous task given the fact that ontology construction is a highly purpose-dependent complex procedure. A very exciting fact in this regard is that as ontologies are built for specific purposes, comparing an ontology with another one (even if they are in the same domain but built for different purposes) is not useful. Secondly, to save time we first decided to re-use any existing ontology in the business domain, but an extensive search through the literature and online databanks and lists of available ontologies produced negative results. Therefore we decided to build our own ontology which will be specific to our purpose. Extensive details of how ontologies were built from news articles and other sources will be set out in chapter 9. It will be shown that exploiting term-relationships can be very useful in building domain ontologies,

⁴We refer to bins as a folder where we keep documents, so for company bins, documents are put in individual folders for the companies. We also create temporal bins where we put the news articles according to the publication time, which is called Base Time-line or **BT** or **DateLine**

3.1.6 Analyzer

The analyzer is the subtask where the last and major attention needs to be focused. In this thesis, we will describe two explanatory models. The first is the **keyword-based model** for explaining an event, which finds referential keywords which are indicative of the possible explanation of an event. The second model is a **sentence-based model** which actually tries to find explanatory sentences from the huge information source, which can show why a certain event, such as a stock price-drop, happened. Though they seem to be very similar approaches, but when we went deep in finding possible models for explanatory sentences, we found that the useful methods are very very different from the keyword-based approaches.

3.1.6.1 Keyword-based Model

In the keyword-based model we try to explain any major event with a set of keywords extracted from news articles. In this model the information theoretic approach was used and produced a good result. This method is applied on the artificial market with real money (IEM political markets), the artificial market with artificial money (Foresight exchange) and on the real market with real money (stock exchange). Important keywords and phrases found, indicate the real life events for the price changes.

3.1.6.2 Sentence-based Model

In the sentence-based model, explanatory sentences indicative of possible reasons for any major event are provided. In this model we used a mixture of a learning model along with the traditional relevance model with newly developed concept of language model. One of the results of this research is that the latter model can perform even better than the keyword-model and can be more useful in pinpointing the reason for any major event.

At the end a functional and behavioural comparisons between these two approaches will be provided and the advantages and disadvantages of these models and ways in which they be improved will be identified. Hopefully these models can be used not just for finding explanations of past events, but also for learning and thus for providing forecasts for stock price movements and more. While it is a type

of open problem to forecast stock price dynamics, we hope that these methods can play a crucial role in that respect.

Chapter 4

Getting the Right Information: Feature-based Extraction of Informative Blocks from Web-pages

“Information is a source of learning. But unless it is organized, processed, and available to the right people in a format for decision making, it is a burden, not a benefit.”

- William Pollard

(American Nuclear Physicist, Writer, Author, Administrator, Priest)
(1912-1989)

In this chapter the text-filtering approach which was developed is elaborated. An obvious question here is: why we focus on a data preparation stage so deeply for the removal of redundant information. The primary reason is to get the right information for the data analysis. Without the proper and correct information no algorithm, regardless of how efficient it is, can find the explanation for an event. However, one of the off-the-shelf text extractors could have been used. Actually, as will be shown in this chapter, re-implementations of the recent best extractor algorithms could not provide sufficiently good and acceptable results.

Applications of our analyser algorithm on the results of these extractors could not give acceptable keyword explanations. Therefore we decided to improve the performance of the overall system by introducing our own algorithms [44, 46, 47].

4.1 Motivation

Web-pages – especially dynamically generated ones – contain several items that cannot be classified as the “primary content”, e.g., navigation sidebars, advertisements, copyright notices, etc. Most clients and end-users search for the primary content, and largely do not seek the non-informative content. A tool that assists an end-user or application to search and process information from Web-pages automatically must separate the “primary content sections” from the other content sections. Both types of these sections are referenced to as “Web-page blocks” or just “blocks”. First, a tool must segment the Web-pages into Web-page blocks and second, the tool must separate the primary content blocks from the non-informative content blocks. In this work, Web-page blocks are formally defined and a new algorithm is devised to partition an HTML page into constituent Web-page blocks. Four new algorithms are then proposed: *ContentExtractor*, *FeatureExtractor*, *K-FeatureExtractor*, and *L-Extractor*. These algorithms identify primary content blocks by (i) looking for blocks that do not occur a large number of times across Web-pages, (ii) looking for blocks with desired features, and (iii) using classifiers, trained with block-features, respectively. While operating on several thousand Web-pages obtained from various Websites, the algorithms in this study outperform several existing algorithms with respect to runtime and accuracy. Furthermore, it will be shown that a Web-cache system that applies our algorithms to remove non-informative content blocks and to identify similar blocks across Web-pages can achieve significant storage savings.

4.2 Introduction

Search engines crawl the World-Wide Web to collect Web-pages. These pages are either readily accessible without any activated account or they are restricted by username and password. In whatever way the crawlers access these pages, they

are (in almost all cases) cached locally and indexed by the search engines.

An end-user who performs a search using a search engine is interested in the *primary informative content* of these Web-pages. However, a substantial part of them – especially those that are created dynamically – is content that should not be classified as the primary informative content of the Web-page. These blocks are seldom sought by the users of the Website. We refer to such blocks as *non-content blocks*. Non-content blocks are very common in dynamically generated Web-pages. Typically such blocks contain advertisements, image-maps, plug-ins, logos, counters, search boxes, category information, navigational links, related links, footers and headers, and copyright information.

Before the content from a Web-page can be used, it must be subdivided into smaller semantically-homogeneous sections based on their content. Such sections are referred to as *blocks* in the rest of the paper. A block (or Web-page block), $\mathcal{B}$, is a portion of a Web-page enclosed within an open-tag and its matching close-tag, where the open and close tags belong to an ordered tag-set $\mathcal{T}$ that includes tags like $\langle \text{TR} \rangle$, $\langle \text{P} \rangle$, $\langle \text{HR} \rangle$, and $\langle \text{UL} \rangle$. Figure 4.1, shows a Web-page obtained from CNN’s Website¹ and the blocks in that Web-page.

In this work, the problem of identifying the primary informative content of a Web-page is addressed. From the empirical observations, it was found that approximately three-fourths of the dynamically generated pages found on the Web, have a table in them. An HTML table is defined using the tag $\langle \text{TABLE} \rangle$. In a table occurring in a Web-page, each cell is considered to be a block. Where tables are not available, identifying blocks involves partitioning a Web-page into sections that are coherent, and that have specific functions. For example, a block with links for navigation is a navigation block. Another example is an advertising block that contains one or more advertisements that are laid out side by side. Usually, a navigation block is found on the left side of a Web-page. Typically, the primary informative content block is laid out to the right of a Web-page. Four algorithms, *ContentExtractor*, *FeatureExtractor*, *K-FeatureExtractor*, and *L-Extractor*, were designed and implemented for this study, which identify the primary content blocks in a Web-page.

An added advantage of identifying blocks in Web-pages is that if the user does

¹<http://www.cnn.com>

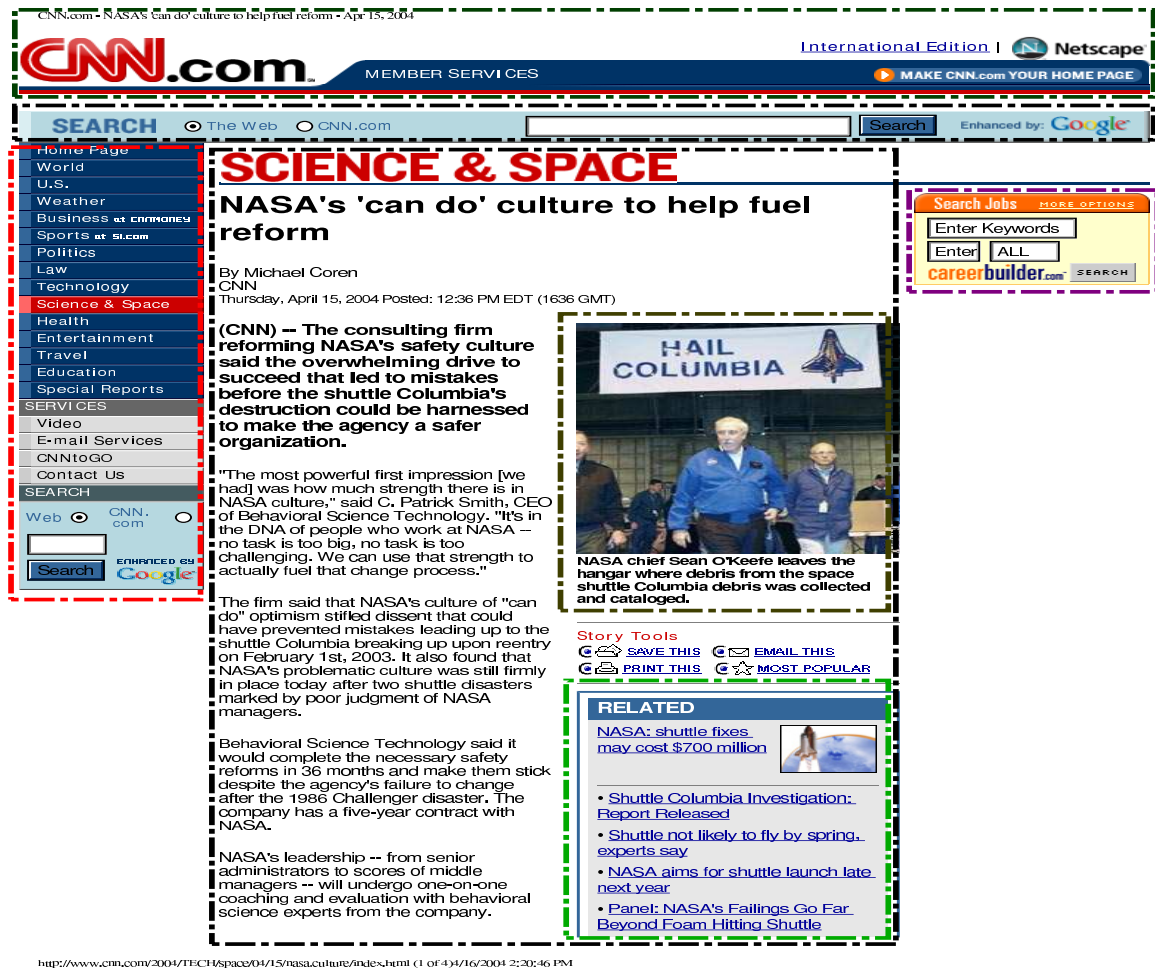


Figure 4.1. A Web-page from CNN.com and its blocks (shown using boxes)

not require the non-content blocks or requires only a few non-content blocks, the rest of the blocks can be deleted. This contraction is useful in situations where large parts of the Web are crawled, indexed and stored. Since the non-content blocks are often a significant part of dynamically generated Web-pages, eliminating them results in significant savings with respect to storage cache and indexing.

The study can identify similar blocks across different Web-pages obtained from different Websites. For example, a search on Google News on almost any topic returns several syndicated articles. Popular items like syndicated columns or news articles written by global news agencies like AP or Reuters appear in many newspapers. Even the top 100 results returned by Google contain only very few unique

columns related to the topic because of duplicates published at different sites. Ideally, the user wants only one of these several copies of articles. Since the different copies of the article are from different newspapers and Websites, they differ in their non-content blocks but have similar content blocks. By separating and indexing only the content blocks, one can easily identify that two Web-pages have identical content blocks, save on storage and indexing by saving only one copy of the block, and make search results better by returning more unique articles. Even search times improve because there is less data to search.

To identify and separate content blocks from non-content blocks four simple yet powerful algorithms, called *ContentExtractor*, *FeatureExtractor*, *K-FeatureExtractor*, and *L-Extractor* are proposed. Different types of blocks are characterized based on the different features they possess. *FeatureExtractor* is based on this characterization and uses heuristics based on the occurrence of certain features to identify content blocks. *K-FeatureExtractor* is a special modification of *FeatureExtractor* which performs better in a wide variety of Web-pages. *ContentExtractor* identifies non-content blocks based on the appearance of the same block in multiple Web-pages. *L-Extractor* uses various block-features and train a Support Vector (SV) based classifier to identify a informative block vs. a non-informative block.

First, the algorithms partition the Web-page into blocks based on heuristics. These heuristics are based on our previous study of HTML editing style over a few thousand Web-pages. Lin and Ho [84] have proposed an entropy-based algorithm that partitions a Web-page into blocks on the basis of HTML tables. In contrast, this study considers not only HTML tables but also other tags, combined with the heuristics to partition a Web-page. Secondly, these algorithms classify each block as either a content block or a non-content block. While the algorithm decides whether a block, $\mathcal{B}$, is content or not, it also compares $\mathcal{B}$ with stored blocks to determine whether $\mathcal{B}$ is similar to a stored block. Both *(K-)FeatureExtractor* and *ContentExtractor* produce excellent precision and recall values and runtime efficiency and above all, do not use any manual input and require no complex machine learning process. *L-Extractor* is still under experimentation, but it seems to produce fairly high accuracy.

While operating on several thousand Web-pages obtained from news and various other Websites, these algorithms significantly outperform their nearest com-

petitor, the Entropy-based blocking algorithm proposed by Lin and Ho [84]. The *ContentExtractor* algorithm was also compared with the Shingling algorithm devised by Ramaswamy et.al. [113, 114]. *ContentExtractor* achieves similar savings on storage requirements as the Shingling algorithm. However it outperforms the Shingling algorithm significantly with respect to runtime, showing that simple heuristics can suffice to identify primary content blocks in Web-pages.

The rest of this chapter is organized as follows: Section 4.3 covers the related work. The concept of “blocks” and a few related terms are described in section 4.4. The algorithms are described in sections 4.5, 4.6, and 4.8. Section 4.7 outlines the performance evaluation plan and the data set on which the experiments were run. This study’s algorithms are compared with the *LH* and Shingling algorithm in subsection 4.7.4. Section 4.9 set out conclusions and future work.

4.3 Related Work

Yi and Liu [133, 85] have proposed an algorithm for identifying non-content blocks (they refer to it as “noisy” blocks) of Web-pages. Their algorithm examines several Web-pages from a single Website. If an element of a Web-page has the same style across various Web-pages, the element is more likely than not to be marked as a non-content block. Their algorithm also looks at the entropy of the blocks to determine non-content blocks. Their technique is intuitively very close to the concept of “information content” of a block. This is one of the very innovative ideas examined in this study.

The algorithms in this work, only look at the inverse block document frequency (defined below) and features of blocks. In order to identify the presentation styles of elements of Web-pages, Yi and Liu’s algorithm constructs a “Style Tree”. A “Style Tree” is a variation of the DOM sub-structure of Web-page elements. If there are Web-pages whose elements have the same style but different contents and yet are non-content blocks, the algorithms in the current work would not be able to detect that. However, in practice, these algorithms, even in the presence of advertisement images that vary from page to page, can identify such images as non-content blocks by making use of the text in the blocks that are almost the same. Since these algorithms use simple heuristics to determine non-content

blocks, this process does not incur the overhead of constructing “Style Tree”s.

Another work that is closely related is that of Lin and Ho [84]. The algorithm they proposed also tries to partition a Web-page into blocks and to identify content blocks. They used the entropy of the keywords used in a block to determine whether the block is redundant. The current study has a more comprehensive definition of blocks and demonstrates that the algorithm herein designed and implemented gives better precision and recall values than Lin and Ho’s algorithm as shown below.

Cai, et.al. [27] have introduced a vision-based page segmentation (VIPS) algorithm, which segments a Web-page based on its visual characteristics, identifying horizontal spaces and vertical spaces delimiting blocks much as a human being would visually identify semantic blocks on a Web-page. They use this algorithm to show that better page segmentation and a search algorithm based on semantic content blocks improves the performance of Web searches. Song, et.al. [125] have also used VIPS to find blocks in Web-pages. Then they use Support Vector Machines (SVM) and Neural Networks to identify important Web-pages. The current study observed that VIPS is significantly more expensive than the proposed simple blocking algorithm. The first step of (*L-Extractor*) uses the blocking algorithm and the second step uses a SVM-based algorithm to achieve good results. *ContentExtractor* and *k-FeatureExtractor* use even simpler and less expensive techniques to identify primary content blocks in Web-pages.

Ramaswamy, et.al., [113, 114] propose a Shingling algorithm to identify fragments of Web-pages and use it to show that the storage requirements of Web-caching are significantly reduced. It is shown below that a *ContentExtractor*-based algorithm provides similar savings for Web-caching, and is at the same time *ContentExtractor* is significantly less expensive than the Shingling algorithm.

Bar-Yossef and Rajagopalan [22] have proposed a method to identify frequent templates of Web-pages and pagelets (identical to the blocks as described in the current study). Yi and Liu argue that their entropy-based method supersedes the template identification method. It will be shown that the method developed in the current work produces better result than the entropy-based method.

Kushmerick [81, 80] has proposed a feature-based method that identifies Internet advertisements in a Web-page. It is solely geared towards removing advertise-

ments and does not remove other non-content blocks. While their algorithm can be extended to remove other non-content blocks, its efficacy for the general Web-cleaning problem has not been studied. Besides, their algorithm generates rules from training examples using a manually-specified procedure that states how the features to be used can be identified. This manual specification is dependent upon applications. The algorithms in this study do not require any manual specification or training data set (except *L-Extractor*).

There has been substantial research on the general problem of extracting information from Web-pages. Information extraction or Web mining systems try to extract useful information from either structured or semi-structured documents. Since a large percentage of dynamically-generated Web-documents have some form of underlying templates, Wrapper [81, 80], Roadrunner [35], Softmealy [73] and other systems try to extract information by identifying and exploiting the templates. Systems like Tsimmis [29] and Araneus [21] depend on manually provided grammar rules. Information Manifold [78, 83], Whirl [32], or Ariadne [19] attempted to extract information using a query system that is similar to database systems. In Wrapper systems [81], the wrappers are automatically created without the use of hand-coding. Kushmerick et.al. [81, 80] have found an inductive learning technique. Their algorithm learns a resource's wrapper by reasoning about a sample of the resource's pages. In Roadrunner [35], a subclass of regular expression grammar (UFRE or Union Free Regular Expression) is used to identify the extraction rules by comparing Web-pages of the same class and by finding similarities or dissimilarities among them. In Softmealy [73], a novel Web-wrapper representation formalism has been presented. This representation is based on a finite-state transducer (FST) and contextual rules, which allow a wrapper to wrap semistructured Web-pages containing missing attributes, multiple attribute values, variant attribute permutations, exceptions and typos, features that no previous work can handle. A SoftMealy wrapper can be learned from labelled example items using a simple induction algorithm. For other semi-structured wrapper generators like Stalker [93], a hierarchical information-extraction technique converts the complexity of mining into a series of simpler extraction tasks. It is claimed that Stalker can wrap information sources that can not be learned by existing inductive learning techniques. Most of these approaches are geared toward learning the regular ex-

pressions or grammar induction [30] of the inherent structure or the semi-structure and so computational complexities are quite high.

The efforts mentioned above are involved in extracting information that originally came from databases. This underlying data stored in databases is very structured in nature. This work concentrates on Web-pages in which the underlying information is unstructured text. The techniques used for information extraction are applied on entire Web-pages, whereas they actually seek information only from the primary content blocks of the Web-pages.

Using the current study’s algorithm to extract the primary content blocks of the Web-pages as a pre-processing step, and then running the information extraction algorithms on the primary content blocks reduces the complexity and increase the effectiveness of the extraction process.

Preliminary work [44] shows great improvements in extracting the informative blocks from Web-pages. The feature-based algorithm can be enhanced by using machine learning mechanisms to select the useful features that are used to identify the non-content blocks. The study of using the Support Vector Learning approach in this context is described in section 4.8.

4.4 Segmenting Web-pages into Blocks

This section defines the concept of “blocks” in Web-pages and a few other related terms. Most Web-pages on the Internet are still written in HTML [33]. Even dynamically generated pages are mostly written with HTML tags, complying with the SGML format. The layouts of these SGML documents follow the Document Object Model tree structure of the World Wide Web Consortium ². Out of all these tags, Web authors mostly use <TABLE> to design the layouts. Our algorithm uses <TABLE> as the first tag on the basis of which it partitions a Web-page. After <TABLE>, it uses <TR>, <P>, <HR>, , <DIV> and etc. as the next few partitioning tags in that order. The order of the tags was selected based on observations of Web-pages and a belief that it is the natural order used by most Web-page designers (according to a study of HTML editing style for a few thousand Web-pages from various sources and formats). For example, <TABLE>

²W3C or <http://www.w3c.org>

comes as first partitioning tag since we see more instances of `<UL>` in a table cell than `<TABLE>`s coming inside `<LI>`, an item under `<UL>`. This study's algorithms partition a Web-page based on the first tag in the list to identify the blocks, and then sub-partitions the identified blocks based on the second tag and so on. It continues to partition until there is no tag left in a block in the block-set which is part of the list of tags. This ensures that the blocks are atomic in nature and no further division is possible on them. The partitioning algorithm is illustrated in subsection 4.5.1 and this tag-set is called the partitioning tag-set.

4.4.1 Block Features

By definition, blocks may include other smaller blocks. But in the current implementation, as described above, only atomic blocks have been used for computation purposes (except in a few cases in *FeatureExtractor*). Atomic blocks will have features like text, images, applets, javascript, etc. Actually, all HTML tags (Following W3C (<http://w3c.org>)) except the tags in a partitioning tag-set are included for feature analysis. A *block-feature set* is a set of features that a block contains. Several features are associated with their respective standard tags but not all features have standard tags. For example, an image is always associated with the tag `<img>`; however, the text feature has no standard tag. For features that are associated with a tag, the W3C guidelines on HTML pages were used to make the full list of features. The features of a block used include, but are not limited to, Text, Text-tag, List, Table, Link, Object, Frame, Form, Script, Style-Sheet, etc. The most important (and nice) quality of the algorithm is that this list can be updated as time and version of HTML pages change, without making any fundamental changes in the algorithm.

Examples of individual features in the feature vectors constructed by this algorithms are: the number of terms (in case of text feature), the number of images (in case of `<IMG>` tag), the number of javascripts (in case of `<SCRIPT>` tag), etc. However, for text blocks, simply taking the number of terms in the block may result in falsely identifying two blocks as similar. Therefore, the features are augmented by the addition of a binary feature for each term in the corpus of documents. If a term occurs in a block, the entry in the corresponding feature vector is a one,

otherwise it is zero. If other features are deemed important, this framework can be easily modified by the addition of new features and adjustment in the weights of those features while computing the similarity between blocks.

4.4.2 Inverse Block Document Frequency and Block Similarity

ContentExtractor computes the Inverse Block Document Frequency (*IBDF*) as defined below. For example, if a block appears in multiple Web-pages, say, in most of CNN's Web-pages, the block will have a smaller Inverse Block Document Frequency (*IBDF*) than one that appears only in one Web-page.

Let us assume $IBDF^i$ represents the *IBDF* of a block $\mathcal{B}_i$ in a set of pages $\mathcal{S}$. Typically, the set $\mathcal{S}$ consists of similar pages from the same source. $IBDF^i$ is inversely proportional to the number of Web-pages the block $\mathcal{B}_i$ occurs in.

So $\mathcal{S}$ is a set of Web-pages of the same class, i.e., obtained from the same source. Then

$$\mathcal{S} = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \dots, \mathcal{P}_M\}. \quad (4.1)$$

where $\mathcal{P}_i$ s ($\forall i \in M$) are individual HTML pages from that source.

And in

$$IBDF^i \equiv f\left(\frac{1}{|\mathcal{S}^i| + 1}\right) \quad (4.2)$$

where

$$\mathcal{S}^i = \cup\{\mathcal{P}_l : Sim(\mathcal{B}_i, \mathcal{B}_k) < \epsilon, \forall \mathcal{B}_k \in \mathcal{P}_l, \forall \mathcal{P}_l \in \mathcal{S}\} \quad (4.3)$$

f denotes a function, usually a linear or log function. The function $Sim(\mathcal{B}_i, \mathcal{B}_k)$ is a similarity measure of the two blocks. An expert provides the threshold ϵ .

There may be a question regarding whether the basis of this algorithm is a rule-based technique. Actually an analogy is drawn between the TF-IDF measure in vector-space model [118] and the *IBDF* measure. As the commonly occurring or redundant words or phrases in a collection are eliminated by applying the IDF measure of all the words and phrases, the same concept is extended for blocks. If we consider blocks as the atomic units in a Web-page, it is easy to visualize that the blocks having lower *IBDF* values or high frequency of occurring in several Web-pages will be eliminated as redundant blocks. TF-IDF measure and related

algorithms are undoubtedly not rule-based algorithms. Likewise *IBDF* measure and *ContentExtractor* should not be considered as rule-based approaches.

4.5 Algorithm: ContentExtractor

The input to the algorithms is a set (at least two) of Web-pages belonging to a *class* of Web-pages. A class is defined as a set of Web-pages from the same Website whose designs or structural contents are very similar. A set of Web-pages dynamically generated from the same script is an example of a class. The output of the algorithms are the primary content blocks in the given class of Web-pages.

The first step of all the algorithms in the current study is to use the *GetBlockSet* routine (described next) to partition each page into blocks.

4.5.1 GetBlockSet

The *GetBlockSet* routine takes an HTML page as input with the ordered tag-set.

GetBlockSet takes a tag from the tag-set one by one and calls the *GetBlocks* routine for each block belonging to the set of blocks, already generated. New sub-blocks created by *GetBlocks* are added to the block set and the generating main block (which was just partitioned) is removed from the set. The *First* function gives the first element (tag) of an ordered set, and the *Next* function gives the consecutive elements (tags) of an ordered set.

4.5.2 GetBlocks

GetBlocks takes a full document or a part of a document, written in HTML, and a tag as its input. It partitions the document into blocks according to the input tag. For example, in the case of the `<TABLE>` tag given as input, it will produce the DOM tree with all the table blocks. It does a breadth-first search of the DOM tree (if any) of the HTML page. If the input tag is `<TABLE>` and there is no table structure available in the HTML page, it does not partition the page. In that case the whole input page comes back as a single block. In the case of other tags such as `<P>`, it partitions the page/block into blocks/sub-blocks separated by those tags. Figure 4.2 shows the structure of two HTML pages. It also shows

the blocks that our blocking algorithm identifies for each of these pages (under the dotted line).

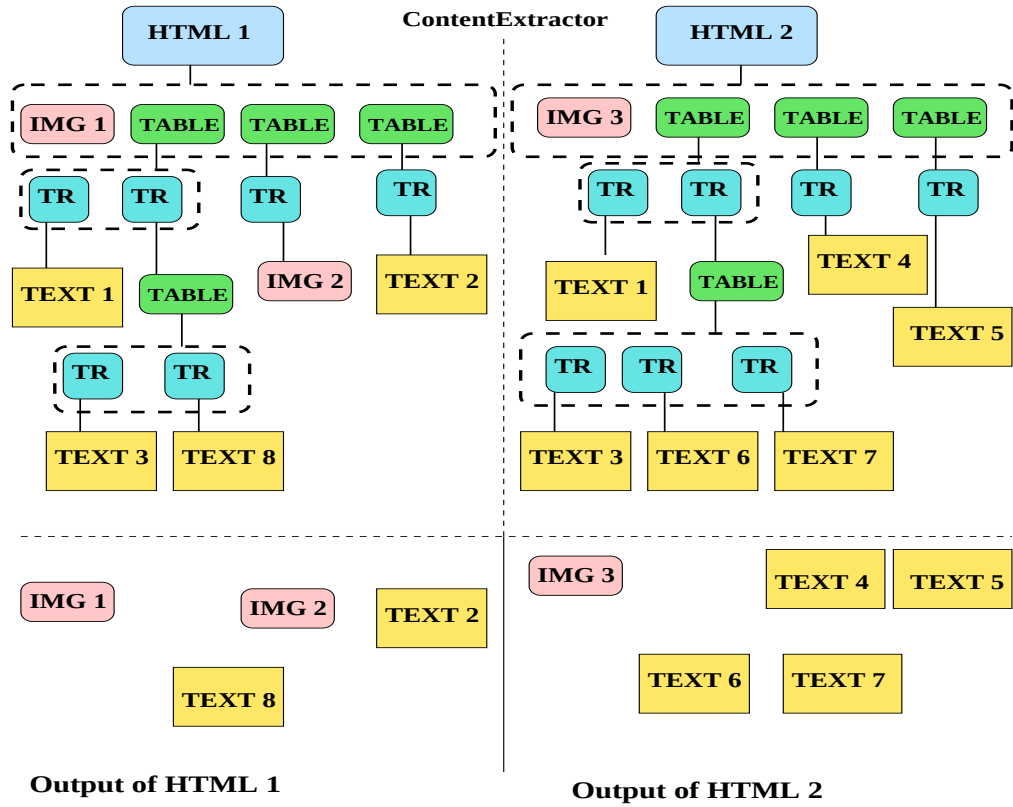


Figure 4.2. Two Web-pages' block structures as seen by *GetBlockSet*. The output from them are shown under the dotted line.

4.5.3 Identifying Primary Content Blocks

After the blocks have been identified, the second step of the process involves identifying the primary content blocks and separating them from the non-content blocks. All four algorithms identify the primary content blocks in the Web-pages, but their methodologies are different.

4.5.4 ContentExtractor

We show the pseudo-code for *ContentExtractor* in Algorithm 1. It calculates the *IBDF* values of each block. For implementation purpose we compute the *IBDF* values as a counter, and compare with θ , which is same as comparing $IBDF^{-1}$ with θ^{-1} . The algorithm used a similarity measure function *Sim*, to find out the similarity between two blocks.

4.5.4.1 The Similarity Function and Threshold

Given two blocks, *Sim* returns the cosine between their block feature vectors. We used a threshold value of $\epsilon = 0.9$. That is, if the similarity measure is greater than the threshold value, then the two blocks are accepted as identical. The threshold value can be changed according to the needs of the application and affects the precision and recall of the algorithm. Blocks that occur rarely across different Web-pages, i.e., have low *IBDF*'s are output as the primary content blocks.

4.5.4.2 Complexity Measure

The computational complexity of this approach is dependent on the computation of the similarity measure between blocks and the computation of the *IBDF*'s of the blocks.

Let us assume there are N blocks per page and the total number of documents in a class is M . According to the definition of a class, these pages are derived from the same script or from the same Website. In practical cases, pages derived from the same class are of the same design. Their headers, left panels, or the footers are similar (depending on the threshold ϵ). Thus, during the comparison, the number of completely new blocks coming from the second page is pretty low. Therefore,

```

Input : Set  $\mathcal{S}$  of HTML pages, Sorted tag-set  $\mathcal{T}$ 
Output: Primary Content Blocks and their associated pages in  $\mathcal{S}$ 
begin
   $\mathcal{M}_{BD} \leftarrow \emptyset$ 
  { Here the  $\mathcal{M}_{BD}$  matrix is the block-document matrix where rows
  represent document and columns represent block identifier.}
  for each  $H^k \in \mathcal{S}$  do
    { Here  $B^k$  represents the  $k$ th row of the  $\mathcal{M}_{BD}$  matrix. }
     $B^k \leftarrow \text{GetBlockSet}(H^k, \mathcal{T})$ 
     $\mathcal{M}_{BD}^k \leftarrow B^k$ 
  end

  for each  $b_{ij} \in \mathcal{M}_{BD}$  do
     $IBDF_{ij}^{-1} \leftarrow 1$ 
    for each  $b_{kl} \in \mathcal{M}_{BD}$  do
      { Here  $i \neq k$ .}
       $Sim_{ijkl} \leftarrow Sim(b_{ij}, b_{kl})$ 
      if  $Sim_{ijkl} > \epsilon$  then
         $IBDF_{ij}^{-1} \leftarrow Update(IBDF_{ij}^{-1})$ 
        {Update Recalculates  $IBDF^{-1}$ }
      end
    end
  end

  { If  $IBDF^{-1}$  value above threshold we will produce the output }
  for each  $b_{ij} \in \mathcal{M}_{BD}$  do
    if  $IBDF_i^{-1} > \theta^{-1}$  then
      | Output the content of the block
    end
  end
end

```

Algorithm 1: *ContentExtractor*

when the algorithm compares pages P_i and P_{i+1} , we can arguably assume that the similar blocks will largely outnumber the dissimilar blocks.

Suppose the average number of new blocks in $P_i + 1$ that are not present in P_i is κ . Then from above discussion $\kappa \ll N$. Accordingly, after the first comparison, a $(N + \kappa) \times M$ dimensional Block-Document matrix will be formed. The computational complexity of this step is $O(N^2)$. After these pages are compared, the blocks

```

(Function GetBlockSet : )
Input  : HTML page  $H$ , Sorted tag-set  $\mathcal{T}$ 
Output: Set of Blocks in  $H$ 
begin
   $B \leftarrow H$ ; // set of blocks, initially set to H.
   $f \leftarrow \text{Next}(\mathcal{T})$ 
  while  $f \neq \emptyset$  do
     $b \leftarrow \text{First}(B)$ 
    while  $b \neq \emptyset$  do
      if  $b$  contains  $f$  then
         $B^N \leftarrow \text{GetBlocks}(B, f)$   $B \leftarrow (B - b) \cup B^N$ 
      end
       $b \leftarrow \text{Next}(B)$ 
    end
     $f \leftarrow \text{Next}(\mathcal{T})$ 
  end
end

```

Algorithm 2: *GetBlockSet Function (ContentExtractor Algorithm continued)*

```

(Function Sim : )
Input  :  $\text{Block}_1, \text{Block}_2$ 
Output: Similarity Measure
begin
  //FeatureVector produces a vector of all
  //features as enlisted and described above
   $\overline{\mathcal{F}}_1 \leftarrow \text{FeatureVector}(\text{Block}_1)$ 
   $\overline{\mathcal{F}}_2 \leftarrow \text{FeatureVector}(\text{Block}_2)$ 
  return  $\cos(\overline{\mathcal{F}}_1, \overline{\mathcal{F}}_2)$ 
end

```

Algorithm 3: *Sim Function (ContentExtractor Algorithm continued)*

of the third page will be compared with the combined set of blocks coming from first two pages. When the second step of the comparison will be performed, the cost of computation will be increased. Ultimately the total number of comparison

will be

$$\begin{aligned}
& N^2 + (N + \kappa) \times N + (N + 2\kappa) \times N + \dots + (N + (M - 2)\kappa) \times N \\
&= (M - 1)N^2 + \frac{\kappa}{2}(M^2 - 3M + 2)N \\
&= (MN^2 - N^2 + \frac{\kappa}{2}M^2N - \frac{3\kappa}{2}MN + \kappa N) \\
&= O(M^2N)
\end{aligned} \tag{4.4}$$

as $M \gg N$ and $\kappa \ll N$.

The Block-Document matrix computation will be dependent on the value of M or the number of pages in the set and the average number of blocks in each individual page. In the future, we would like to explore if taking a smaller number of pages in a set is enough for identifying the irrelevant blocks. The time complexity to make the sorted block-document matrix is $O(M^3N^2\log(N))$.

If all Web-pages in the same class are dynamically generated from the same template, and running *ContentExtractor* for all M documents is excessively costly, in practice, we can identify the template using fewer than M documents. Then, for all M documents, the primary content block that appears at a fixed place in the template can be extracted using the template.

4.6 Algorithm: FeatureExtractor /K-FeatureExtractor

We now show our second algorithm, *FeatureExtractor*. We designed *FeatureExtractor* such that any informative block (corresponding to any feature) can be identified. For example, *FeatureExtractor* invoked with the features text, image, links, identifies the text blocks, image blocks or navigational blocks as the primary content blocks respectively. We show the pseudo-code for *FeatureExtractor* in Algorithm 4.

4.6.1 Block Features

The following list describes the features of a Web-page block that we have used in our implementation. A Web-page block can have any or all features of an HTML page. The W3C HTML guidelines have been followed here.

- Text : The text content inside the block.
- Text-tag: The text tags, e.g., <h1>, <h2> etc. inside the block.
- List : The lists available inside the block.
- Table : Available tables inside the block.
- Link : URLs or links inside the block.
- Object : Image, Applet etc. available in the block.
- Frame : Frames inside the block. Usually it is rare to have frame in the block, but to make the list complete it has been added.
- Form : Forms available inside the block.
- Script : Javascripts or other types of scripts written in the block.
- Style-Sheet : This is also to make the list complete and compliant to W3C guidelines. Styles are usually important for browser rendering, and usually included inside other tags, like links and tables etc.

A question may arise here that why we are taking these features. As aforesaid, all these blocks are HTML blocks and we are trying to find out a particular block or set of blocks which can be identified by the block-property such as text-blocks or image-blocks. In FeatureExtractor we are looking for all the text-blocks and so we need to compare the properties of a block against other blocks. These comparison is only possible if we consider all the HTML tags as the feature-set of the blocks. As we mentioned earlier, we can update this list if we so desire because of changes in HTML features or because of an application's updated preferences of desirable features easily without fundamentally changing the algorithm.

Unlike the *ContentExtractor* algorithm, the *FeatureExtractor* algorithm does not depend on multiple Web-pages but depends on the feature-set and the chosen feature for output. The set features are HTML features as explained before. For example, let us consider the chosen feature is text ($\mathcal{T}_I$). Now our algorithm calculates a value for each feature in each block. Say, a block contains 1000 words and 2 images and 3 links and an applet, and the maximum values of words, images, links, and applets contained in blocks in the data-set are 2000, 4, 50 and 3. Then the values for the features in the given block are $1000/2000$, $2/4$, $3/50$, and $1/3$ respectively. After that we put each block in the winner-basket if the sum of the feature values of the desired features is greater than the sum of the feature values of the rest of the features. From this winner-basket, we recompute the feature values for this new set of blocks, and chose the one with highest value of desired feature.

Now according to this algorithm a block with a single word and nothing else would be the obvious winner and will be chosen. In most practical cases this scenario did not arise. And also, we do not consider a single row or column of a table as a block. We consider the whole table (in the highest depth of table tree) as a block. So the chance of getting a block with a single word is distant.

4.6.2 K-FeatureExtractor

Though *FeatureExtractor* performs with high precision and recall for one of our datasets, it may not do so in general and can be improved. For Web-pages with multiple important text blocks, a typical reader may be interested in all the sections not just one of them (winner of *FeatureExtractor*). For example, an end-user may be interested in all the reviews available from a page in Amazon.com and each review is in a separate block. General shopping sites, review sites, chat forums etc. may all contain multiple blocks of important textual information. *FeatureExtractor* shows poor precision and recall as it produces only one text-block with highest probability, while other important blocks are not retrieved. To overcome this drawback, we revised the last part of the *FeatureExtractor* and named the new algorithm as *K-FeatureExtractor* (Algorithm 5). To handle more general Web-pages of varied editing-styles, we improved the *FeatureExtractor* algorithm.

```

Input   : Set of HTML pages  $H$ , Sorted Tag Set  $\mathcal{T}$ , Desired Feature  $\mathcal{F}_I$ 
Output : Content Blocks of  $H$ 
Feature: Feature set  $\mathcal{F}_S$  used for block separation sorted according to im-
           portance taken from  $\mathcal{T}$ 

begin
   $B \leftarrow GetBlockSet(B, \mathcal{T})$ 
  {  $\mathcal{W}$  is the output variable that records potential output block sets }
   $W \leftarrow \emptyset$ 
  for each  $b \in B$  do
     $P_1 \leftarrow Pr(\mathcal{F}_I | \mathcal{F})$ 
     $P_2 \leftarrow Pr((\mathcal{F} - \mathcal{F}_I) | \mathcal{F})$ 
    if  $P_1 > P_2$  then
       $W \leftarrow W \cup b$ 
    end
  end

  { Now depending on the condition or choice we will produce output
  from the set  $\mathcal{W}$  }
  for each  $b \in \mathcal{W}$  do
     $P_b \leftarrow Pr(\mathcal{F}_I | \mathcal{F}, \mathcal{W})$ 
    //  $\mathcal{F}_I = \mathcal{T}_I$  in the experiment
  end
  { Output: Sort  $\mathcal{W}$  according to the Probability value  $P_b$  and (1)
  Produce the content of the Winner block }
end

```

Algorithm 4: FeatureExtractor

Instead of taking just the winner block from the winner-basket, we apply a k-means clustering algorithm to select the best probability blocks from the basket. This helps us get high precision and recall from shopping Websites and review Websites and in general a much broader range of Websites. The results from using the *K-FeatureExtractor* for these types of Web-pages are shown in table 4.3 separately. Needless to mention that *FeatureExtractor* did not do well for these Web-pages. *K-FeatureExtractor* uses an adaptive K-means clustering on the winner set to retrieve multiple winners as opposed to *FeatureExtractor* that selects a single winner. The usual values of k taken are 2 or 3, and the initial centroids are chosen from the sorted list at equidistant index values. After the clustering is done, the

high probability cluster(s) are taken and the corresponding text contents of all those blocks are taken as the output.

```

begin
  ... same as FeatureExtractor except the last statement ...
  { Output: Sort  $\mathcal{W}$  according to the probability value  $P_b$  and (2) Use
    k-means clustering and take high probability cluster(s). Combine the
    text contents from all of the blocks. }
end

```

Algorithm 5: K-FeatureExtractor

4.7 Experimental Evaluation

In this section, we present empirical evaluation of our methods. We also compare our algorithms with two other major competitors.

4.7.1 First Comparison: With the LH Algorithm

We implemented and compared our algorithm with *LH*, the entropy-based algorithm, proposed by Lin and Ho [84]. They use the terms *precision* and *recall* to refer to the metrics to evaluate their algorithm. Although, the use of these terms are somewhat different from their usual sense in the field, “Information Retrieval”, in order to avoid confusion, we use the same terms (added with a “b-” for blocks) to refer to the evaluation metrics of our work.

4.7.2 Metric Used

Precision is defined as the ratio of the number of relevant items (actual primary content blocks) r found and the total number of items (primary content blocks suggested by an algorithm) t found. Here we used a block level precision and so we call it as *b-Precision*.

$$b - Precision = \frac{r}{t}. \quad (4.5)$$

Recall has been defined as the ratio of the number of relevant items found and the desired number of relevant items. The desired number of relevant items includes

<i>Site</i>	<i>Address</i>	<i>Category</i>	<i>Number</i>
ABC	http://www.abcnews.com	Main Page, USA, World, Business, Entertainment, Science/Tech, Politics, Living	415
BB	http://www.bloomberg.com	Main Page, World, Market, US Top Stories, World Top Stories, Asian, Australia/New Zealand, Europe, The Americas	510
BBC	http://www.bbc.co.uk	Main Page, The Continents, Business, Health, Nature, Technology, Entertainment	890
CBS	http://www.cbsnews.com	Main Page, National, World, Politics, Technology, Health, Entertainment	370
CNN	http://www.cnn.com	Main Page, World, US, All Politics, Law, Tech(nology), Space (Technology), Health, Showbiz, Education, Specials	717
FOX	http://www.foxnews.com	Main Page, Top Stories, Politics, Business, Life, Views	476
FOX23	http://www.fox23news.com	Main Page, General, Local, Regional, National, World, In Depth, Sports, Business, Entertainment, Health	658
IE	http://www.indianexpress.com	Main Page, International, Sports, National Network, Business, Headlines	269
IT	http://www.indiatimes.com	Main Page, Main Stories, Top Media Headlines	454
MSNBC	http://www.msnbc.com	Main Page, Business, Sports, Technology and Science, Health, Travel	647
YAHOO	http://news.yahoo.com	Main Page, Top Stories, US (National), Business, World, Entertainment, Sports, Technology, Politics, Science	505
Shopping	http://www.shopping.com	Miscellaneous Products	100
Amazon	http://www.amazon.com	Book Pages	100
Barnes And Noble	http://www.bn.com	Book Pages	100
Epinion	http://www.epinions.com	Reviews	100

Table 4.1. Details of the dataset. The number of pages taken from individual categories are not shown due to the enormous size of the latex table, but the interested reader can contact authors to get the details.

Site	b-Prec of LH	b-Recall of LH	b-F- measure of LH	b-Prec of CE	b-Recall of CE	b-F- measure of CE	b-Prec of (K-) FE	b-Recall of (K-) FE	b-F- measure of (K-)) FE
ABC	0.811	0.99	0.89	0.915	0.99	0.95	1.00	1.00	1.00
BB	0.882	0.99	0.93	0.997	1.00	0.998	1.00	1.00	1.00
BBC	0.834	0.99	0.905	0.968	1.00	0.983	1.00	1.00	1.00
CBS	0.823	1.00	0.902	0.972	1.00	0.985	0.98	0.977	0.978
CNN	0.856	1.00	0.922	0.977	1.00	0.988	0.98	0.98	0.98
FOX	0.82	1.00	0.901	0.967	1.00	0.983	1.00	0.99	0.994
FOX23	0.822	1.00	0.902	0.985	1.00	0.992	1.00	1.00	1.00
IE	0.77	0.95	0.85	0.911	0.993	0.95	0.93	0.99	0.959
IT	0.793	0.99	0.878	0.924	0.981	0.951	0.96	0.98	0.969
MSNBC	0.802	1.00	0.89	0.980	1.00	0.989	0.92	1.00	0.95
YAHOO	0.730	1.00	0.84	0.967	1.00	0.98	1.00	0.95	0.974

Table 4.2. Block level Precision and Recall values from *LH* algorithm, *ContentExtractor* and *FeatureExtractor*. The second, third, and fourth columns are from *LH* algorithm, the fifth, sixth, and the seventh columns are from *ContentExtractor* and the eighth, ninth, and tenth columns are from *(K-)FeatureExtractor*. We put *K* in parenthesis to imply that these results are almost same for *FeatureExtractor* and *K-FeatureExtractor*

the number of relevant items found and the missed relevant items m . In case of blocks we call it as block level recall or $b - Recall$.

$$b - Recall = \frac{r}{r + m}. \quad (4.6)$$

Similar to the way it is defined in information retrieval literature by Van Rijsbergen [116], we can refer to the F-measure here as the $b-F-measure$ and define it as

$$b - F - measure = \frac{2 * (b - Precision) * (b - Recall)}{(b - Precision) + (b - Recall)} \quad (4.7)$$

4.7.3 Data Set

Exactly like Lin and Ho, we chose several Websites from the news domain. We crawled the Web for news articles and other types of Websites to collect documents. The details (name, source, category, number) of the dataset are shown in Table 4.1.

In total we took 15 different Websites including news, shopping, opinion posting Websites etc. whose designs and page-layouts are completely different. In table 4.2, we took 11 different news Websites for first comparison. Unlike Lin and Ho's

dataset [84] that is obtained from one fixed category of news sections (only one of them is “Miscellaneous” news from CDN), we took random news pages from every section of a particular Website. This choice makes the dataset a good mix of a wide variety of HTML layouts. This step was necessary to compare the robustness of their algorithm to ours.

4.7.4 Performance Comparison

We implemented all four algorithms in Perl 5.8.0 on a Pentium-based Linux platform. With the generous help from a few graduate students and professors, we calculated the b-precision and b-recall values for each Website and layout category for text feature. These values are shown in tables 4.2 and 4.3.

Our algorithms outperform *LH* in all news sites in all categories. The b-recall is always good since all algorithms could find most relevant blocks but the results obtained by running the *LH* algorithm were less precise than those obtained by *ContentExtractor* since the former algorithm also includes lots of other non-content blocks.

We believe that the primary reason for the poor b-precision of *LH* is because of the greedy approach taken by their algorithm while identifying the solution. A second reason is that the *LH* algorithm works at the feature level instead of the block level. *LH* gives high redundancy score to features that occur across Web-pages. The redundancy score of a block is proportional to the weighted sum of the redundancy scores of each feature it contains. Instead of looking at occurrences of features across Web-pages, the *ContentExtractor* algorithm looks at occurrences of similar blocks across pages. This fundamental difference results in better b-precision obtained by our algorithm.

The *FeatureExtractor* algorithm only works well on Web-pages where the primary Web-pages have one dominant feature. For example, in news Web pages, text is the dominant feature. However, if we go to a domain where the primary content is a mix of multiple features, *FeatureExtractor*’s b-precision suffers. If *FeatureExtractor* has to be deployed in such a domain, it must be modified to handle multiple features and use a weighted measure of the presence of multiple features to identify the primary content pages. Due to the dependence of *FeatureExtractor*

Site	b-Prec of LH	b-Recall of LH	b-F- measure of LH	b-Prec of CE	b-Recall of CE	b-F- measure of CE	b-Prec of K-FE	b-Recall of K-FE	b-F- measure of K- FE
Shopping	0.79	1.00	0.88	0.971	1.00	0.985	1.00	0.99	0.994
Amazon	0.771	0.99	0.86	0.98	1.00	0.989	1.00	0.967	0.983
Barnes And Noble	0.81	1.00	0.895	0.982	0.98	0.98	1.00	0.968	0.983
Epinion	0.79	1.00	0.88	0.97	1.00	0.984	1.00	0.956	0.977

Table 4.3. Block level Precision and Recall values from *LH* algorithm, *ContentExtractor* and *FeatureExtractor*. The second, third, and fourth columns are from *LH* algorithm, the fifth, sixth, and the seventh columns are from *ContentExtractor* and the eighth, ninth, and tenth columns are from *K-FeatureExtractor*. Due to poor performance of *FeatureExtractor* for these Web-pages (which we do not show here) we improved it to *K-FeatureExtractor*

on one particular feature, we expect it to perform poorer than *ContentExtractor* in more general cases where the dominant features in a primary content block are not known.

In other cases (the last four Websites in table 4.1) where there is a single dominant feature but multiple blocks should be in the winner set (not just a single winner), *FeatureExtractor* may not perform well. And supporting our intuition, *FeatureExtractor* resulted in poor performance for these Websites. Because of that, we used *K-FeatureExtractor* for these Websites. The results are shown in table 4.3 compared to our *ContentExtractor* and *LH* algorithms.

4.7.5 b-Precision and b-Recall

Both *FeatureExtractor* and *ContentExtractor* performed better than *LH* in almost all cases. Actually with *ContentExtractor*, there are few or almost no missing blocks because the algorithm discards only repetitive blocks and keeps the other blocks and repetitive blocks have low real information content. In the news domains, most primary content blocks were dominated by text content and so *FeatureExtractor* deployed with the mandate to find blocks with predominantly text content, performs well. The b-precision of *ContentExtractor* increases with the number of pages involved in *IBDF* calculation. We compare the features of the

Property	LH	ContentExtractor	(K-)FeatureExtractor
b-Precision	<i>Low</i>	<i>High</i>	Very High
b-Recall	<i>High</i>	<i>Very High</i>	Very High
Number of pages needed	<i>All the pages to calculate Entropy of features</i>	<i>Very few (5–10) pages from same class are enough to give high performance</i>	A single HTML page is all that is needed
Time of completion	<i>Always more than ContentExtractor</i>	Less than LH (shown in figure 4.3)	Even less than ContentExtractor

Table 4.4. A property-wise comparison table for three algorithms. Note that *(K-)FeatureExtractor* here represents both *FeatureExtractor* and *K-FeatureExtractor*. For the case of *(K-)FeatureExtractor* we took the b-precision, b-recall, b-F-measure and all other comparisons with respect to text feature.

first three algorithms in Table 4.4.

4.7.6 Execution Time

Figure 4.3 shows execution time taken by the three algorithms (*LH*, *ContentExtractor* and *FeatureExtractor*) averaged over all test Web-pages. We did not include *K-FeatureExtractor* as the time taken by it would be same and will overlap the lowermost curve to make it more cluttered. From the figure it is clear that our algorithms outperform the *LH* algorithm by a significant margin. We can further increase the performance of *ContentExtractor* by generating a template of a set of Web-pages using five to ten Web-pages from a site and using the template to extract primary content blocks.

Here in table 4.4 we present a comparison table for the features of both algorithms. This table shows the clear difference between *LH* and our *ContentExtractor* and *(K-)FeatureExtractor* algorithms.

4.7.7 Second Comparison: With Shingling Algorithm

In this section, we compare one of our algorithms with the Shingling algorithm proposed by Ramaswamy et.al. [113]. Regarding the way this algorithm is designed, it is the closest to our *ContentExtractor* algorithm, and therefore we will attempt

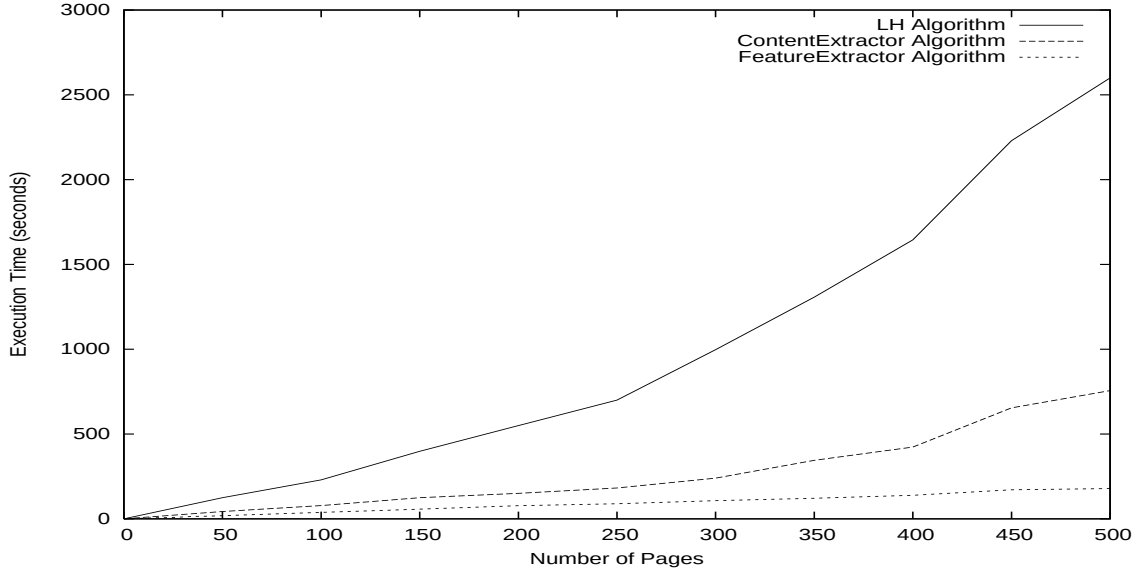


Figure 4.3. Run-times for the *LH*, *ContentExtractor* and *FeatureExtractor* algorithms. The vertical axis represents the time of execution (in seconds) for a number of pages (plotted in the horizontal axis).

to compare these two algorithms side-by-side. They also partition the HTML page into blocks (in their case they call them the nodes of the AF or Augmented Fragment tree). Then they characterize each individual node with different properties, such as *SubtreeValue*, *SubtreeSize*, *SubtreeShingles* and others. The detection of similar nodes was done by an algorithm called “Shared Fragment Detection”. The Shingling algorithm was designed to save storage for Web-caches. First, we compare the storage requirements of a Web-cache using a *ContentExtractor* algorithm versus one obtained by the Shingling algorithm. Table 4.4 shows a comparison of the Shingling algorithm with *ContentExtractor*. To show the amount of storage-savings obtained, we show the initial storage requirement of few HTML files, when neither of our algorithms have been run. It is evident that both *ContentExtractor* and the Shingling algorithm provide substantial (and almost similar) amount of savings in caching size.

Precision and recall values using *Shingling* algorithm are very very close (from table 4.6) to our those from *ContentExtractor* algorithm as we see from the Web-pages we have analysed. Thus, the main advantage of *ContentExtractor* is its time

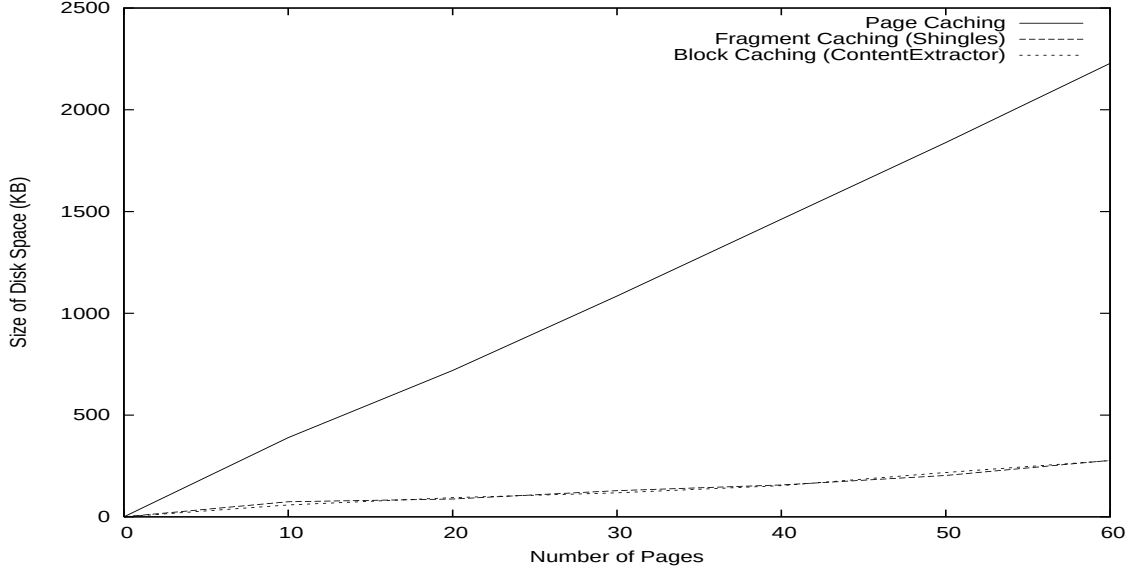


Figure 4.4. Total storage requirement of Web-caches using *ContentExtractor*-based and Shingling algorithms.

of execution. Due to less complex steps and easy feature-based characterization of individual blocks in *ContentExtractor* the generation of informative blocks is very fast. The *Shingling* algorithm depends mainly on calculating the hash values of all $(N - W + 1)$ possible token-IDs for N tokens and shingles set of window length W . The computation of hash values and the computation for the comparison involved in the resemblance equation

$$Resemblance(A_i, B_j) = \frac{SubtreeShingles(A_i) \cap SubtreeShingles(B_j)}{SubtreeShingles(A_i) \cup SubtreeShingles(B_j)} \quad (4.8)$$

are expensive. *ContentExtractor* does not remove any HTML tags and uses them for making the feature vector. This computation is relatively simple and inexpensive because the comparison/similarity is based on just a cosine calculation between two vectors. Therefore, *ContentExtractor* is much faster than the Shingling algorithm. Figure 4.5 shows a comparison of run-times taken by Shingling algorithm and *ContentExtractor*. Clearly, the Shingling algorithm does not scale very well and thus, the times for the larger number of Web-pages is not reported.

Property	Shingling Algorithm	ContentExtractor
Atomic Structure	AF Tree Node	Block
Basis of Similarity	Shared Fragment Measure	IBDF Measure
Similarity Calculation	ShareFactor	IBDF value
Similarity Threshold	ShareFactor	θ^{-1}
Matching	MinMatchFactor	ϵ
Atomic Property	Complex and expensive measurement of $(N-W+1)$ node-ids.	Simple HTML tag feature, which is perfect for measuring similarity. Inexpensive.
Precision and Recall	Similar to ContentExtractor	Similar to Shingling
Speed	Slow	Much faster
Disk Space Requirements (Figure 4.4)	Similar to ContentExtractor	Similar to Shingling

Table 4.5. A property-wise comparison table for *Shingling algorithm* and *ContentExtractor*

Site	b-Prec of Shingling	b-Recall of Shingling	b-F-measure of Shingling	b-Prec of CE	b-Recall of CE	b-F-measure of CE
ABC	0.92	0.99	0.953	0.915	0.99	0.95
BB	0.98	1.00	0.989	0.997	1.00	0.998
BBC	0.971	1.00	0.985	0.968	1.00	0.983
CBS	0.97	0.99	0.979	0.972	1.00	0.985
CNN	0.97	1.00	0.984	0.977	1.00	0.988

Table 4.6. Block level Precision and Recall values from *Shingling* algorithm, and *ContentExtractor* algorithm for 50 Web-pages. The second, third, and fourth columns are from *Shingling* algorithm, the fifth, sixth, and the seventh columns are from *ContentExtractor*.

4.8 Algorithm: L-Extractor

Song et.al. [125] used VIPS to perform page segmentation and then used an SVM to identify primary content blocks in a Web-page. VIPS is an expensive page segmentation algorithm. However, we hypothesized that an SVM can be very useful to identify primary content blocks. To prove our hypothesis, we applied our *GetBlockSet* algorithm to 250 Web-pages. In the next step we created the feature-vectors for these blocks using HTML tags as described above. This set includes all HTML tags except that are included in the partitioning-list of tags and including text feature. We then ran our Support Vector Learning classifier [72] (We used

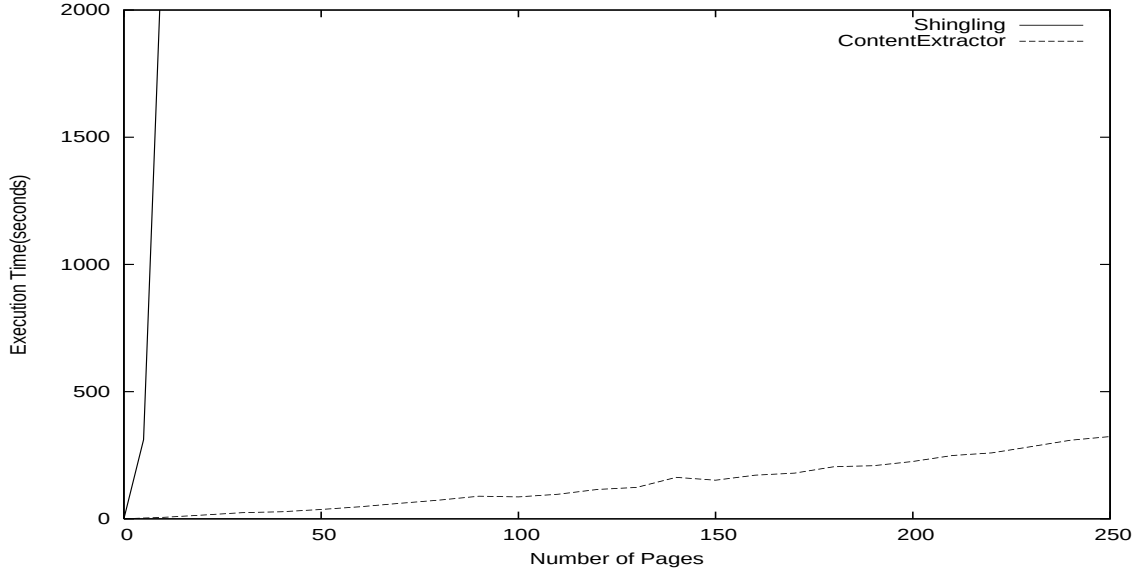


Figure 4.5. Total execution time taken for *ContentExtractor* and the Singling-based algorithm.

a linear Kernel for the Perl SV classifier with cost and weight values of two-class C-SVC algorithm both set to 1 with 5-fold cross validation following Chang and Lin [74]). Figure 4.6 shows the accuracy of finding the informative blocks over increasing number of Web-pages. From this study we can claim that our block-partitioning algorithm combined with an SVM works with high efficiency.

4.9 Conclusions

Simple, yet powerful, and modular algorithms were devised to identify primary content blocks from Web-pages. These algorithms outperformed the LH algorithm significantly, in b-precision as well as run-time, without the use of any complex learning technique. The *FeatureExtractor* algorithm, provided a feature, can identify the primary content block with respect to that feature. The *ContentExtractor* algorithm detects redundant blocks based on the occurrence of the same block across multiple Web-pages. The algorithms, thereby, reduce the storage requirements, make indices smaller, and result in faster and more effective searches. Though the savings in filesize and the precision and recall values from “Shingling

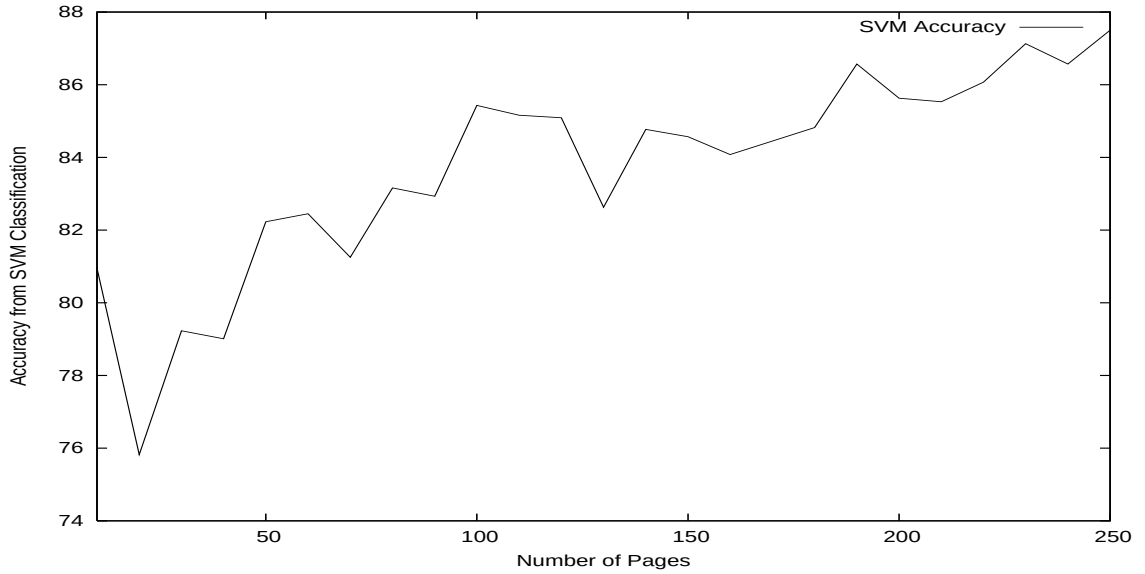


Figure 4.6. Accuracy obtained by using our blocking algorithm with an SVM-based classifier based on block features to identify primary content blocks.

Algorithm” is as good as from *ContentExtractor*, *ContentExtractor* outperforms the “Shingling Algorithm” by a high margin in run-time. The next step would be to deploy these algorithms as a part of a system that crawls Web-pages, and extracts primary content blocks from them. As part of this process, primary content would be determined and heuristic algorithms will be sought for to identify the semantics of the content to generate markup. The storage requirement for indices, the efficiency of the markup algorithms, and the relevancy measures of documents with respect to keywords in queries should also improve (as shown briefly by caching size benefit) since now only the relevant parts of the documents are considered.

Now, as the text content is in hand, a further step would be to identify the temporal information from them. As explained earlier “why something happened” can only be answered if “when that happened” is known. So getting temporal information and associating the articles with that information is crucial to proceed with the agenda. In the next chapter the algorithm to extract the time of publication of a news article will be introduced. This time of publication has been named as base time-line or **BT** of the article (the reason will be explained in the next chapter itself). A support vector learning classifier based approach was introduced

and it was ascertained that a right set parameter can produce a good performance in finding the base time-line.

Chapter 5

Right Information at Right Time: Temporal Ordering

*"Everybody gets so much information all day long that they lose their
common sense."*

- **Gertrude Stein**
(American Writer)
(1874-1946)

After getting the correct information, or more precisely, the clean information from Web pages, how to associate them with the proper time-lines is crucial. The proper time-line of a news article is its date and time of publication. There are other deeper levels of temporal associations, which we plan to use in the final work, but for the time being the basic need is to find the base time-lines of the articles. In future we plan to extract all the events from the article and will try to associate them with the individual time-lines of their contexts. To do that we need to extract the available temporal information for those events, and then try to find out the time-lines with the help of a base time-line. More on this will be elaborated soon, but before that let us discuss the motivation behind it again [45].

5.1 Motivation

The importance of textual information manifolds when it can be associated with the actual time-line of when it happened. This is more true with the on-line publishing of news articles for a particular subject. We introduce a novel approach to tag these news articles with temporal information, whenever available. This involves a temporal baseline, which needs to be established for the entire article. Temporal baseline is defined as the date and possibly time of when the article had first been published, as stated in the article. Without a precise and correct temporal baseline, no further processing can be possible. To accurately find the temporal baseline we approached with a support vector based classification method. We found that the proper choice of parameters to train the support vector classifier can result in high accuracy. We showed the data collection phase, training phase, and testing phase and report our accuracies for web-pages from 26 different Web sites. From this accuracy we can claim that our approach can be used to find the temporal baseline very accurately.

5.2 Introduction

In Web data mining, retrieving relevant documents has always been of great importance. This is related to other important areas of research such as text summarization and question answering. All these areas of research requires situating a document in proper time-line for better precision and recall and this has formed a major research area of temporal information processing. Lots of efforts have recently been put to tag news articles with temporal information and great strides have been made using areas such as rule-based and natural language based techniques. Researchers like Hwang and Schubert, Kemp and Reyle, Lascarides and Asher, Allen, Hitzman, Kehler and others have used knowledge sources, tense, aspect, adverbs, rhetorical relations and of course background knowledge. For example, Lascarides and Asher used “narration” relation in sentences to identify the time of events. Others have found that news stories may not be a right place to use the narrative convention. As researchers found, events in news articles are tough to order. But even before starting to order the events in a news article, the first

and foremost requirement is to find out the sentences carrying any occurrences of time units such as year, month, week, day and so on.

In most cases these time units are relative. Which means that these time units are not expressed in complete time unit formats ¹ This makes temporal ordering difficult in news articles. We have to fill up the missing parts according to the current notion or context of time in the article. For example, if month and date are available and year is not mentioned, then we can fill up the gap using the year when the news article was published. If time and day are mentioned, we can try to fill up the month and year similarly. We consider these time lines, where any part is missing, as relative time-lines, as they are relative in time with the context of the article. Therefore, in almost all cases, the missing parts can be understood from the temporal baseline. We refer temporal baseline or base time-line interchangeably.

Base time-line is defined as the date and time (if available) of the publication of the news article, *as mentioned in the article itself*. Almost all news articles mention the date of publication somewhere inside the body. It may not be the exact same time when the news article came on-line (file update date and time) or the time of its crawling (crawl date and time). To properly identify other relative date and time expressions in the article and to calculate the exact date and time, it is very important to find out the base time-line (henceforth called **BT**). In this work, we consider only those articles which has base time-line. An article without any mention of its publication date or time can not be covered by this approach of us. For any referenced temporal expression (e.g. “last Friday”, “next month”, “next weekend” etc.) to be tagged with other temporal information such as year, month, possible date and time, it is obvious that we need to add or subtract the proper difference of time from the **BT** of that article. Therefore to order the events of a news article using temporal information we need to find the **BT** of the article, and then find the relative time-lines of individual events in the article. In this chapter we concentrate on finding the **BT** of an article.

The problem is not trivial due to several reasons. Firstly, temporal expressions are not used in a standard way. It has many different forms due to different languages. In a particular language people can express the dates and times in

¹Complete time units are usually expressed in **YY::MM::DD::HH::mm::SS** following the ISO8601 guidelines or at least in a similar way, which can easily be converted to ISO8601 format using simple converter algorithms.

many different ways too. So finding temporal expressions is not trivial. It needs a proper grammar. We have developed our own grammar. We do not focus on the grammar part for a few reasons. Firstly to claim the superiority of the performance of our grammar we needed to compare its performance with other grammars or techniques. Unfortunately in this area of research, we found it is hard to get grammars used by other researchers, or their implementation. Secondly it was not our focus here to build a better grammar. A moderately accurate and workable grammar is enough for the task. We are more interested in finding the **BT**. **BT** is very crucial in temporal ordering and yet we found that almost every researcher assumed that this information is available somehow or other. Unfortunately that is not true. Due to the on-line information explosion, it is very difficult to keep up with the publishing speed of all news articles regularly and to crawl and cache them in timely manner. The natural delay in crawling, caching and putting them in proper temporal bins brings the necessity of an algorithm which is very necessary and important to find the **BT** of these articles. Researchers in this area almost always assume that this publication date is available or they take the first available date in their implementations, which is not the right choice as we will see soon. We approached the problem by identifying the proper set of parameters by which we will train a learning classifier. Agreeing with our intuition, our choice of parameter-set combined with the support vector based classifier, produced high accuracy.

The rest of the chapter is organized as follows. Some of the related previous efforts in this area of research are mentioned in the next section. In section 5.4 we describe our approach, in section 5.5 we mention the time format and standards, used in the experiment and in section 5.6 we describe the method to prepare the data. After that in section 5.7 we evaluated the classification accuracy and conclude thereafter.

5.3 Related Work

Tagging news article with temporal information has received much attention these days. Most of the prior work is based on natural language research. Unfortunately, we have not seen much prior work to find the base time-line of an article. The

reason, as explained above, is due to the limited assumption that data can be available sorted according to the date and time of publishing. This is not a workable assumption in today's scenario due to the information explosion in the Web and less time to organize data. In a few instances where we found that researchers had no information about the date of the article, they assumed the first available date as the date of publication of the article. Now it may be true in a very few percentage of article, but we can not rely on this assumption. So actually we could not find any such previous attempt to find the **BT** of an article. We cited a few cases of general temporal information extractors here, some of them are outstanding work.

Starting with Allen's general theory of action and time [16] we have seen very effective efforts towards structuring textual documents into temporally well-defined blocks. Some early approaches are very formal with finding time or time related expressions in documents but they were instrumental in setting up the ground-breaking steps. Based on that others tried to use rule-based or sometimes knowledge-based techniques. But most of the researchers related to text summarization, question answering or temporal ontology building, used or tried to use techniques and advances in Natural Language Processing. Natural language processing has its roots long back in time with Reichenbach [115] who pointed out the difference between the point of speech (time of utterance), the point (time) of the event and point of reference or the reference time. In Time Frames [79], Koen and Bender stated the benefits of the time augmentation of news and approaches the problem with time or time-related information already available in the content. Their time extractor extracts time with moderate precision and recall. MIT's Questioning News System [117] used individual documents of a set, but did not create a temporal structure as such. Kullberg, in his masters thesis created a visualization tool to see dynamic time-lines in historical events. Other researchers such as Allen [17], Dorr [55], Mani [87], Lascarides [82], Passonneau [99], Ferro [57], tried to approach it from Natural Language Processing perspective using discourse structures, tense of the verb or the aspect. But as we have seen and explained before there is not enough evidence of classifying the temporal expressions using machine language techniques to find out the base time-line (**BT**) of an article. It sounds obvious that without the proper **BT**, no technique could give the proper

time-line of any events inside the article.

5.4 Our Approach

We used a support vector classifier [72, 122] to find the accurate **BT** of an article. We first used our **TimeFinder** algorithm (based on a temporal grammar described in figure 5.2) to find all possible temporal expressions inside an article. Once it finds and builds the temporal expression set, we compute the values of several parameters (these parameters are described in section 5.6.1) for each of these time expressions. We train our classifier with this data. When we get new articles, we process these articles first through the **TimeFinder** algorithm to generate a set of temporal expressions as we did for the training set. We then measure the parameter values for all these temporal expressions. Then we run our trained classifier to find the **BTs** of these new articles.

5.4.1 Example

Let us show an example page (Figure 5.1). This figure shows a sample HTML page from Yahoo finance ². We put dotted rectangular boxes around the probable time expressions which our **TimeFinder** algorithm is going to extract. From this figure it is clear why we need a learning classifier to classify the **BT** from other temporal expressions. As we see there are lots of temporal expressions in the beginning of this article as surrounded by the dotted rectangular box. But none of them are the base time-line. The base time-line **BT** is the date “December 06, 2002”, which is surrounded by rectangular box just above the heading “**AMERISOURCEBERGEN CORP (ABC)**”. So the first temporal expression is not the publication date of the article as we see here. So clearly we should not be relying on taking the first temporal expression as the **BT**, as taken by lots of previous researchers.

²<http://finance.yahoo.com>


[Search](#) - [Finance Home](#) - [Yahoo!](#) - [Help](#)


SEC Filings
NYSE: ABC

Enter symbol:

[symbol lookup](#)

More Info: [Quote](#) ~ [Chart](#) ~ [News](#) ~ [Profile](#) ~ [Reports](#) ~ [Research](#) ~ **SEC** ~ [Msgs](#) ~ [Insider](#) ~ [Financials](#) ~ [Analyst Ratings](#)

Recent filings: [Aug 14, 2002 \(Qtrly Rpt\)](#) ~ [Aug 14, 2002 \(form8-K\)](#) ~ [Nov 05, 2002 \(form8-K\)](#) ~ [Nov 12, 2002 \(form8-K\)](#) ~ [Nov 13, 2002 \(form8-K\)](#) ~ [Dec 06, 2002 \(form8-K\)](#) ~ [Dec 16, 2002 \(form8-K\)](#) ~ [Dec 24, 2002 \(Annual Rpt\)](#) ~ [Jan 06, 2003 \(form8-K\)](#) ~ [Jan 17, 2003 \(form8-K\)](#) ~ [Jan 24, 2003 \(form8-K\)](#) ~ [Feb 13, 2003 \(Qtrly Rpt\)](#) ~ [Apr 24, 2003 \(form8-K\)](#) ~ [May 12, 2003 \(Qtrly Rpt\)](#) ~ [May 22, 2003 \(form8-K\)](#) ~ [Jun 24, 2003 \(form8-K\)](#) ~ [Jul 24, 2003 \(form8-K\)](#)

More filings for ABC available from [EDGAR Online](#) ~ [Get a Free Trial to Edgar Online Premium](#)

December 06, 2002

AMERISOURCEBERGEN CORP (ABC)

form 8-K

Item 5. Other Events.

On **December 5, 2002**, AmerisourceBergen Corporation (the "Company") issued a press release providing information about an investor meeting held that day by the Company in New York City, providing disclosure of the Company's financial expectations for the fiscal quarter ending **December 31, 2002** and affirming the Company's financial expectations for the fiscal year ending **September 30, 2003**. A copy of the press release is filed as Exhibit 99.1 to this report and incorporated herein by reference.

Recent filings: [Aug 14, 2002 \(Qtrly Rpt\)](#) ~ [Aug 14, 2002 \(form8-K\)](#) ~ [Nov 05, 2002 \(form8-K\)](#) ~ [Nov 12, 2002 \(form8-K\)](#) ~ [Nov 13, 2002 \(form8-K\)](#) ~ [Dec 06, 2002 \(form8-K\)](#) ~ [Dec 16, 2002 \(form8-K\)](#) ~ [Dec 24, 2002 \(Annual Rpt\)](#) ~ [Jan 06, 2003 \(form8-K\)](#) ~ [Jan 17, 2003 \(form8-K\)](#) ~ [Jan 24, 2003 \(form8-K\)](#) ~ [Feb 13, 2003 \(Qtrly Rpt\)](#) ~ [Apr 24, 2003 \(form8-K\)](#) ~ [May 12, 2003 \(Qtrly Rpt\)](#) ~ [May 22, 2003 \(form8-K\)](#) ~ [Jun 24, 2003 \(form8-K\)](#) ~ [Jul 24, 2003 \(form8-K\)](#)

More filings for ABC available from [EDGAR Online](#) ~ [Get a Free Trial to EDGAR Online Premium](#)

EDGAR Online: [Research People in this company](#) ~ [Full text Search](#)

Copyright © 2003 Yahoo! Inc. All Rights Reserved. [Privacy Policy](#) - [Terms of Service](#)

Data and information are provided by [EDGAR Online, Inc.](#) for informational purposes only and not intended for trading purposes. Yahoo! and EDGAR Online, Inc. shall have no liability for the accuracy of the information furnished or for delays, omissions or opinions expressed therein, or for any actions taken in reliance thereon. You may not use the information for any illegal purpose or furnish the information to any person, firm or branch office for commercial re-use or re-sale.

Figure 5.1. A sample Yahoo finance page with temporal expressions highlighted.

```

s    -> ["/",".", "-", ",", ":", ";", "'", "\"", ":", ":", "...] (Separator)
DD   -> [01,02,03,04, ... 31] (Date)
MM   -> [00,01,02,03, ... 12] (Month){Jan=00/01}
YYYY -> [DDDD] (Year){D = [0,1,2, ..., 9]}
SM   -> [Jan, Feb, Mar, ..., Dec] (Short Month Id)
LM   -> [January, February, March, ..., December] (Long Month Id)
SD   -> [Sun, Mon, Tue, ..., Sat] (Short Day Id)
LD   -> [Sunday, Monday, Tuesday, ..., Saturday] (Long Day Id)
NU   -> [one, two, three, four, ... ] (Numbers)
NT   -> [first, second, third, fourth, ... ] (Numbered)
SP   -> [new year, valentine, christmas, Independence, ground-hog,
        st.patrick, st.georges, baptiste, halloween,
        Christmas, Father's day, Mother's day,
        Columbus day, Rosh Hasannah, Yum Kippur,
        Hanukkah, ... ] (Special Days)
HH   -> [01,02,03,04, ... 24] (Hour)
NN   -> [00,01,02, ... 59] (Minute)
SS   -> [00,01,02, ... 59] (Second)
AP   -> [AM, PM, HRS, HR, 'o clock etc.] (Time Designator)

```

etc.

```

Date -> [MMsDDsYYYY, YYYYsMMsDD, SMsDDsYYYY, LMsDDsYYYY, NTsSMsYYYY, ... ]
RefD -> [day before, day before yesterday, yesterday, today, tomorrow, ... ]
RefM -> [last month, present month, this month, next month, ... ]
RefY -> [last year, this year, present year, next year, ... ]
RefT -> [last [NU] second(s), last [NU] minute(s), ... ]
Refh -> [last [HH] hour(s), for [HH] hour(s), next [HH] hour(s), ...]
Refm -> [last [NN] minute(s), for [NN] minute(s),
        next [NN] minute(s), ...]
Refs -> [last [SS] second(s), for [SS] second(s), next [SS]
        second(s), ...]
Time -> [HH AP, HHsNN AP, HHsNNsSS AP, ... ]
etc.

```

Figure 5.2. Our temporal grammar used in the TimeFinder algorithm.

5.5 Time format and Grammar

We need to use a standard date and time format to identify every temporal expression in an article. According to the ISO 8601 guidelines, the standard way of expressing the date is **YYYY-MM-DD** and that of time is **hh:mm:ss**. There is also specifications and specialized off-the-shelf algorithms available for time zone

and day-light saving time data ³

We use our own grammar in TimeFinder algorithm to extract time and date expressions. This include most of the time and date formats like “Jan 20, 2004”, “01/20/2004”, “2004-01-20”, “Jan 20th, 2004”, “20th Jan, 2004” etc. All of our news articles are from US news Web sites and so there was no need of month-date-position disambiguation (“DD/MM/YY” or “MM/DD/YY”) (In future, we would like to show our results in this area. We are using Geographic IP address locator to find out which country the article/story is originating from and with our existing database of the country/ region-wise mapping of date formats we will be able to identify and disambiguate the expressions.) Moreover we also look for phrases like “2 months ago”, “3 weeks after”, “in 5 minutes”, etc. Though we can not always fill every entries of “YYYY-MM-DD hh:mm:ss”, at least to find the **BT** we do not have to worry about that too much. We followed the initial work by Koen and Bender [79] and classify these expressions into the following classes.

- **Interval** Intervals are expressions like “twenty to twenty five minutes” (exactly same examples from Koen and Bender [79]) or “twenty-to-thirty years” etc.
- **Age** Age defines expressions like “2 years old”, “A week after”, “2 months before” etc. In relative sense we can get the time by using the precise base time-line (e.g. “Jan 01, 2004 20:34:00”+A week) equivalent to “Jan 08, 2004 20:34:00” for a workable algorithm.
- **Date** Dates are precise dates such “Jan 02, 2003”, or “03/04/2004” or “03.04.2004” etc. There are various ways of expressing the date as explained above.
- **Precise Time** Precise times are time expressions such “2:00pm”, or “Morning 7’O clock”, or “18:15:01 hours”. All these expressions precisely tell the time of the day. As always we fill-up the missing values with the base value, i.e. a missing second will be replaced by “00”.

³Arthur David Olson and others maintain a database of all current and many historic time zone changes and daylight saving time algorithms: <http://www.twinsun.com/tz/tz-link.htm>

- **Time Duration** “Evening”, “Morning”, “Dawn” etc. are obviously not very precise time expressions, but we can get a clear idea from these expressions in the same way as described in the “Age” part. We can use the base time-line to find out the date and approximate time duration of the event.
- **Special Day** “Christmas”, “New Year’s Eve”, “Thanksgiving”, “Rosh Hashanah” etc. come under this category which can precisely tell the date of the year without any “Precise Time”. Here you need the base year to properly identify the date.

We show a snippet of our TimeFinder grammar in figure 5.2. Due to the space constraint we could not show the whole grammar but we are creating an Website to show the full grammar as well as a CGI script to test the performance of **TimeFinder**. This figure shows a few different ways we can expect the date, time and other temporal expressions’ formats. Creating an extensive grammar is very important to increase the recall value of **TimeFinder**.

5.6 Data Preparation and Training Phase

We measure the values of the parameters (as described below) of all the temporal expressions by using our algorithm **TemporalDataPreparer**. Our algorithm has several passes and it is described here in algorithm 6. **ContentExtractor** is an intelligent HTML to text converter devised by Debnath et. al. ([43]), which breaks the whole page into logical blocks, identifies the redundant un-informative blocks comparing with other HTML pages from the same source and keep the informative blocks of text. During the first pass, the **TimeFinder** function finds all probable temporal expressions in the article (we call this article as “training article”, as it is used to train the classifier), and converts them into ISO 8601 format (as much as possible with unknown fields empty and keeping the other duration/age related expressions as they are). During the second pass, **TemporalDataPreparer** asks the user to identify the **BT** of this training article. It produces all the temporal expressions, generated from the first pass, to the user. The user identifies the correct temporal expression which can be attributed as the **BT** of the article. We

did not use a fancy Web-base interface, but we used a regular linux command line interface for the user input submission ⁴

During the third pass, **TemporalDataPrepaper** pulls every temporal expression hash key and measures the values of different parameters (described next) by using **MeasureParameterValue** function. This function performs all possible counting of paragraphs, sentences, and word occurrences before and after every temporal expression. This needs a complete splitting of the article into paragraphs, sentences and words.

5.6.1 Data Parameters

The following set of parameters are used to create data to train the support vector classifier. We included 15 different parameters to properly characterize the temporal expressions. Most of them fall under distance measures, but some of them can be considered as frequency measures.

- **Paragraph Distance (PD)**: The paragraph distance consists of two parameters – how many paragraphs are there before a time expression or **PDB** and how many paragraphs are there after the time expression or **PDA**.
- **Sentence Distance (SD)**: The sentence distance consists of two parameters – how many sentences are there before a time expression or **SDB** and how many sentences are there after the time expression or **SDA**.
- **Word Distance (WD)**: The word distance also consists of two parameters – how many words are there before a time expression or **WDB** and how many words are there after the time expression or **WDA**.
- **Reporter Names (RN)** Reporter names or the names of reporting agencies (**RN**) are also a prime factor in identifying the beginning of a news story and the time. This idea is very similar to the way we understand the beginning of a story. The distance between each time expression and the names of the reporter or reporting agencies in words or characters are stored. In

⁴We thank the students who helped in building the training and test dataset

```

Input      : HTML Page  $H$ , Parameter Set  $\mathcal{P}$ 
Output    : Training Set to train the Support Vector Classifier
Standard: ISO 8601 standard for date and time
begin
    Extract the textual content from the HTML page
    using our intelligent algorithm, which eliminates
    the redundant blocks such as navigational links,
    headers or footers
     $X \leftarrow \text{ContentExtractor}(H)$ 

    Pass 1:
     $\mathcal{T} \leftarrow \text{TimeFinder}(H)$ 
    Extract all time expressions using our grammar
    Let us assume that the set of all time
    expressions in this page is  $\mathcal{T}$ 

    Pass 2: (User interface)
    Ask the user to specify which time-line  $t \in \mathcal{T}$  is
    the BT.

    Pass 3:
    Measuring the  $\mathcal{P}$  parameter values for the time expressions which are
    selected in the first pass.
    for each  $t_i \in \mathcal{T}$  do
         $P_{t_i} \leftarrow \text{MeasureParameterValue}(t, H, X);$ 
        ( $P_{t_i}$  is a data row in the  $|\mathcal{T}| \times |X| \times |\mathcal{P}|$  matrix.)
    end
    Prepare all parameter values in tabular format and stores them in
    training datafile.
end

```

Algorithm 6: TemporalDataPreparer (for Training phase): This algorithm prepares data to train support vector classifier.

our algorithm, we used a knowledge base of all available reporting agency's names and our algorithm checks the occurrence using some simple regular expression rules and then stores the distance in character from all the temporal expressions to the Reporter names. Sometimes reporter names also come in the middle or end of the article and this may reduce the accuracy, but we believe that with all the other parameters together our approach can find the proper **BT** with high accuracy.

- **Specific Words** Specific words (**SW**) are also very important to properly identify the **BT**. Words like “By”, “On”, etc. has more often been seen near the base time-line’s temporal expression compared to other temporal expressions.
- **Specific Symbols** In the same way we also consider the occurrences and distances between specific symbols (**SS**) and the time expressions. These symbols include special character-set like “-” or “:” which are also common near the base time expression.
- **Font Face Variation** Font face variation (**FFV**) is also another important factor which can be used to identify the location of **BT**. We see that usually the news publication date is placed close to the headline of the news article and usually the headline is written in different character size or in bold face. The regular text in normal font face follows it. Though things are not always written in the same way (that is why it is a challenging problem), yet there is a correlation between their locations and **BT**. We wanted to exploit this correlation and so we marked the places in the document where a change of font face occurs. Then we calculated distance $D^i \forall i \in |\mathcal{T}|$ where D^i is the shortest distance between t_i (the i^{th} temporal expression) and the marks. So if there are M places where font face changes, $D^i = \min (Distance(i, j))$, where $Distance(i, j)$ is character difference between t_i and j^{th} mark.
- **Similarity Measures** Similarity measures involve word level similarity between sentences before and after a time expression. The reason behind choosing this parameter is the observation that usually the headline of a news article and the first paragraph just after the **BT** describe the same event, sometimes even using identical words or phrases.

We have chosen the above parameters to mimic how a human being would find out the **BT** of a news article. Some of the parameters alone may not be sufficient in distinguishing the **BT** from other temporal expressions but we wanted to see the combined effect of learning all these parameters in the context of each individual temporal expressions. Also the question may come as to why we have chosen so many different distance measures e.g. by paragraphs, by sentences, and by

```

Input      : Time expression  $t$ , HTML Page  $H$ , Text Page  $X$  converted from
                $H$ 
Output    : Values of Temporal Data Parameters
begin
  Function MeasureParameterValue( $t, H, X$ )
  We need both  $X$ , and  $H$  as some of the parameter calculations depend
  on HTML characters and some will be calculated from the text version
  of the article.
  begin
    Takes the content  $X$  and breaks it into Paragraphs, Sentences,
    Words ... etc.
     $\mathcal{P}$  is the Data Parameters as described in Section 5.6.1
    for each parameter  $p \in \mathcal{P}$  do
       $p_{t_i} \leftarrow$  value of  $p$  in  $X$  for  $t_i$ ;
      Push  $p_{t_i}$  in  $P_{t_i}$ .
    end
    return  $P_{t_i}$  (A row vector)
  end
end

```

Algorithm 7: MeasureParameterValue (used in both training and testing phase): This function calculates all the parameter values for a temporal expression in an HTML page H .

words. Statistically there are several variations in news article formats. Sometimes only paragraph distance measure is not enough because there may be a redundant paragraph in the HTML page and sometimes there are few paragraphs with several sentences in them and other paragraphs consist of only a single sentence. So taking everything into account will hopefully be helpful.

5.7 Testing and Evaluation Phase

To test and evaluate the effectiveness of our approach we crawled financial news articles from various (here 26) financial Web sites (shown in the table 5.1). These are linked to the corresponding stock symbols in finance pages of Yahoo [75]. From that list, we took 114 stock symbols and their news articles (over 1000 in number) for this experiment. We used half of this set for training and the rest half for testing the classifier. Algorithm 8 shows the testing process.

Site	Avg Para	Avg Sent	Avg Word	Avg Time	Acc
Associated Press	18.71	28.71	494.71	9.57	96.54
Briefing.com	19.07	42.48	677.93	18.54	92.67
BusinessWeek Online	22.78	79.93	1030.65	18.91	93.25
Business Wire	20.72	36.54	585.74	13.33	93.78
CBS MarketWatch	60.10	85.97	929.93	21.14	94.01
CCBN	6.16	9.89	410.77	3.48	95.43
Dow Jones Business News	14.96	21.22	629.34	7.62	93.43
EDGAR Online	56.26	122.25	2005.31	54.92	92.91
EDGAR Online Financials	7.00	12.34	583.60	2.97	94.00
FT.com	14.00	21.00	426.00	8.00	90.22
First Call Events	6.00	14.00	387.00	4.00	91.22
Forbes Magazine	19.33	59.33	925.33	15.33	92.02
Forbes.com	26.04	53.81	696.15	16.96	94.12
Investor's Business Daily	34.75	58.25	779.00	14.50	93.11
Market Wire	16.73	33.40	583.67	10.47	96.68
Morningstar .com	12.62	38.00	650.62	11.00	92.07
Motley Fool	26.11	61.51	832.57	16.39	93.80
NewsFactor	34.12	51.00	725.88	15.75	91.72
PR Newswire	17.59	32.47	560.72	12.68	95.26
PrimeZone Media Network	14.07	25.28	458.21	8.28	96.22
Reuters	14.94	22.57	660.81	8.89	98.60
SmartMoney .com	28.00	72.42	900.92	21.50	92.11
StarMine	8.00	9.72	417.00	3.06	89.03
TheStreet.com	20.73	28.08	303.35	7.39	70.50
Wall Street Transcript	26.67	67.67	1303.33	24.33	95.55
Yahoo	4.20	8.00	423.87	12.30	92.21

Table 5.1. Accuracy values of the output produced by Linear Kernel Support Vector classification. The columns represent **Site** i.e. the source Web sites where the article was published, **Average number of Paragraphs per article**, **Average number of Sentences per article**, **Average number of Words per article**, **Average number of Time Expressions per article**, and **Accuracy**. Here note that sometimes the average number of sentences are really low, but as we randomly check, we realized that sometimes they are long sentences, and sometimes the articles are really short articles as opposed to regular general news articles.

The table 5.1 shows the accuracy of this data. From this table we see that our approach of data preparation and the use of support vector classifier worked pretty well.

```

Input    : Classifier  $\mathcal{C}$ , HTML Page  $H$ 
Output  : Accuracy
begin
   $X \leftarrow \text{ContentExtractor}(H)$ 
  Pass 1:
   $\mathcal{T} \leftarrow \text{TimeFinder}(H)$ 
  Pass 2:
  Measure the  $\mathcal{P}$  parameter values for all the time expressions extracted
  in the first pass.
  for each  $t_i \in \mathcal{T}$  do
     $p_{t_i} \leftarrow \text{MeasureParameterValue}(t_i, H, X)$ ;
     $p_{t_i}$  is a data row in the  $|\mathcal{T}| \times |\mathcal{P}|$  matrix.
  end
  Prepare all parameter values in tabular format and stores them in
  testing datafile.
  Pass 3:
  Feed the testing datafile to the classifier  $\mathcal{C}$ 
  Find the BT and match with the labelled dataset and find the
  accuracy
end

```

Algorithm 8: FindAccuracy (testing phase): This algorithm uses classifier $\mathcal{C}$ to classify the temporal expressions.

5.8 Conclusion

We devised a grammar for temporal expressions, and presented a learning based approach to find the base time-line of a news article. We claimed our contribution in finding the right set of parameters which can efficiently classify the base time-line. We crawled financial news articles from 26 web sites and tested our algorithm. Our algorithm finds the base time-line **BT** with high accuracy. In future we would like to report on associating the base time-line with referenced temporal expressions, disambiguation of temporal expressions due to geographical origins among others. We also tried to use the same technique for other news metadata extraction [41] and preliminar results show promising future.

Chapter 6

Analyzer Model

“Do not go where the path may lead, go instead where there is no path and leave a trail.”

- **Ralph Waldo Emerson**
(American essayist and poet)
(1803-1882)

As described in Chapter 3, **Analyzer** block is the core of the explanation model. This block takes time-aligned news articles and the price change during that time period and produces either a (1) keyword-based explanation or a (2) sentence-based explanation. Based on these directions the functionality of the analyzer model can be classified in two sub-blocks, namely “keyword-based model” sub-block and “sentence-based model”¹ sub-block, as shown in Figure 6.1. In the next two sections the functionality of these sub-blocks is explained briefly and then the next few chapters will be devoted to describing them in detail and evaluating the performance of these models.

¹We will call these model as “keyword-based explanation model”, or “keyword-based model” or keyword model” interchangeably. The same also applies for “sentence-based explanation model”, “sentence-based model” or “sentence model”.

6.1 Keyword-based Explanation Model

The keyword-based explanation model depends mainly on an analysis of keyword characteristics during and before any major event. The date of this major event is named as the “pivotal date”. This is described in the upper portion of Figure 6.1. In all the analysis, the resolution of the time scale is taken as a day. This means that any hourly changes during a day have not been aligned with hourly news articles. Therefore, all the news articles in a day are taken as the unit set of news articles for the day and the day’s trend is measured as the price difference between the day’s price at 16:00 (or 4:00pm) (when the market closes), and the price at 16:01 (or 4:01pm) in the previous day. If the days are indicated as d then price trend ΔP at day d is

$$\Delta P = P_{d16:00} - P_{d+116:01} \quad (6.1)$$

As will be defined in the next chapter, the price values were not taken directly, but rather the log-likelihood values, and the difference between the log-likelihood values of these price values was thresholded.

The news articles or events (in case of deeper temporal processing of individual events) around an event are segregated into two windows, **Window A** (or Window After) and **Window B** (or Window Before).

We examined the effectiveness of bandwidths of these two windows on the overall performance, and we will show that *a variable window bandwidth (W_{A2}) aligned with the price dynamics works better than a fixed one (W_{A1})*. The reasoning and significance behind fixed and variable bandwidths will be described in Chapter 8 and 10, but here fixed and variable windows are defined.

6.1.1 Fixed Window Approach

In the fixed window approach, documents or news articles to be analyzed are taken from fixed interval windows before and after the pivotal date. If the pivotal date is d , W_A^f is the “Window A”, and W_B^f is the “Window B”, they can be defined in the following way.

6.1.1.1 Definition

$$W^f_A = (d_A - d) \text{ where } d_A \text{ is constant and } d < d_A \quad (6.2)$$

$$W^f_B = (d - d_B) \text{ where } d_B \text{ is constant and } d > d_B \quad (6.3)$$

and similarly the document collections are

$$C^f_A = C(d) + C(d_A - d) \quad (6.4)$$

$$C^f_B = C(d - d_B) \quad (6.5)$$

where $C(i)$ means the news article collection for the day i and $C(i - j); i > j$ means all the news articles dated from day j to day i . So for the news articles in “Window A”, all the news articles published on the pivotal day itself plus all the news articles for d_A days ahead are included, while for “Window B”, all news articles of d_B days behind the pivotal date are included.

6.1.2 Variable Window Approach

The variable window approach is a little more complex than the fixed window approach and, as we will see, it works better for real market scenarios. It is more fine-tuned and more subtle. Let us define the “Window A” first. As will be seen from the equation below, though it is named “Window A” or “Window After”, actually it is not like the W^f_A defined above.

6.1.2.1 Definition

$$W^v_A = -(d_A - d) \text{ where } slope(d) \neq slope(d_{(a-1)}), \forall a, A = max(a), \text{ and } d > d_A \quad (6.6)$$

This means that the “Window A” starts from the day d , but goes backward for d_A days as long as $slope(d) = slope(d_A)$. It stops at the first day A when the $slope(d_{(A-1)})$ changes.

$$W^v_B = (d_A - d_B) \text{ where } d_B \text{ is constant and } d_A > d_B \quad (6.7)$$

The document collecting equations are as before. For the “Window A”, all the

news articles during d_A are included as well as the pivotal date.

The keyword-based model has proven to work better under an ontology model. This means that before applying the model algorithm, an ontology-based relevance extraction of the sub-parts (paragraphs or sentences) of the news article provides better results. This fact will be shown in Chapter 10 where the business ontologies for individual companies are used prior to applying the keyword-based model. However in some cases where ontology was used with a fixed window, good results were not produced. These cases will be discussed in Chapter 10. However variable window approach with ontology works better than a fixed window with no ontology.

6.2 Sentence-based Explanation Model

Here the fundamental explanation unit is not the “word” but the “whole sentence”. The sentence-based model is very different from the keyword-based model. The differences are described in Chapter 12. The main differences are the use of a learning based language model and of window bandwidths. Study results show that the sentence-based model has more promising and more helpful outcomes than a keyword-based model.

The sentence-based model is based on learning. This means that past news articles and trend of price dynamics were examined to create general models of price dynamics by aligning the price and the relevant news events inside the news articles. These models are then used for very recent news articles. One very important feature to mention here is that individual models were not created for each company. The models were created completely independent of the companies and these models depend only on the price dynamics. The segregation of the company ontologies and the models is an very bold and important step. In Chapter 12 it will be shown that as the company ontologies are used separately to extract the most relevant information (in the form of sentences) units, the segregation is a necessity.

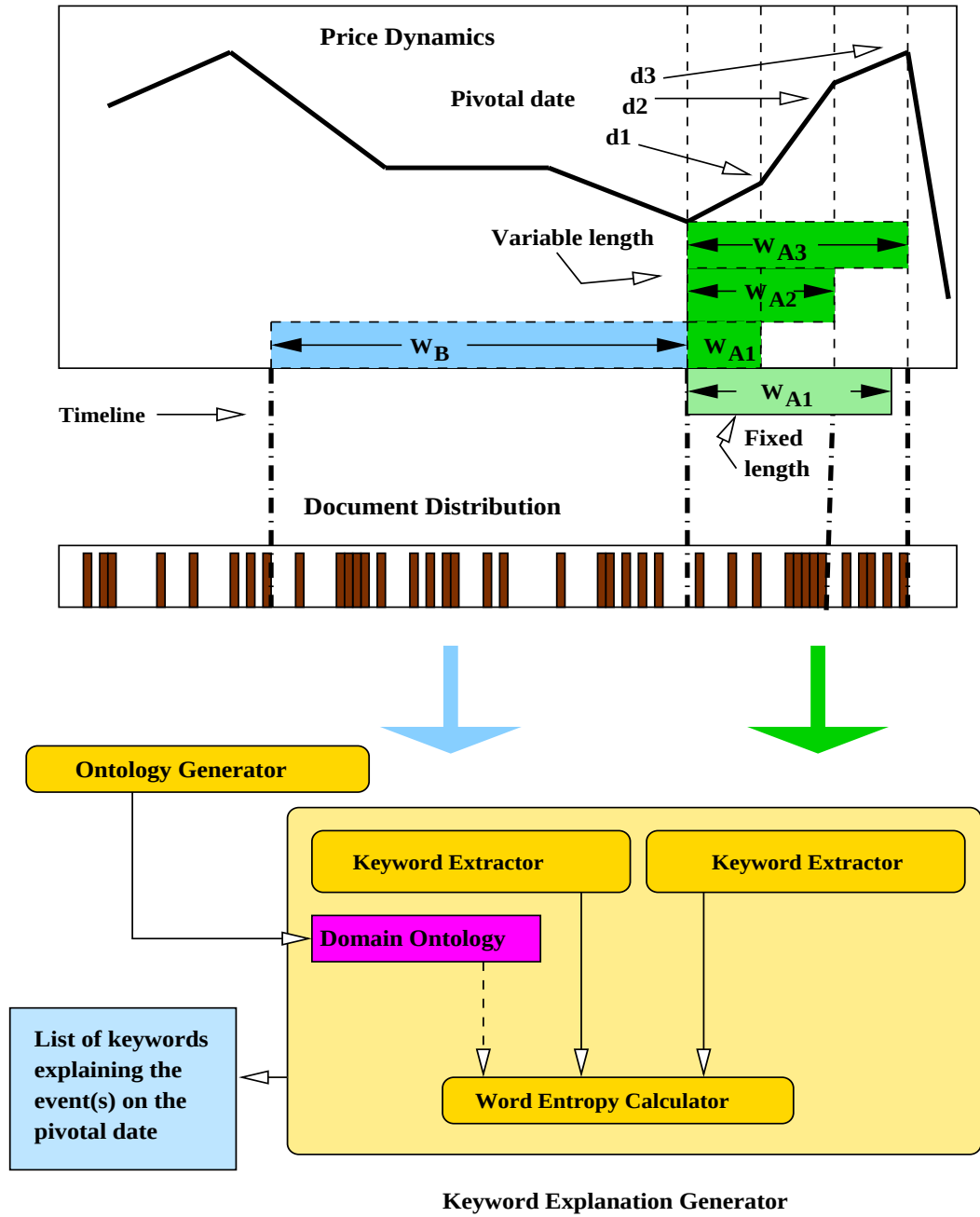


Figure 6.1. Analysis procedure for keyword extraction. The date for which we are trying to find an explanation is called the pivotal date. The documents before the pivotal date are in negative document set or Document B(Before) set. The documents after the pivotal date are put in the positive document set or Document A(After) set.

Chapter 7

Keyword-based Model (Fixed Window)

“I am careful not to confuse excellence with perfection. Excellence, I can reach for; perfection is God’s business.”

- **Michael J. Fox**

(American (Canadian-born) actor)
(1961-)

After the news events are extracted, cleaned, and ordered according to their base time-line **BT** (i.e. publishing date and/or time), they must be analyzed to detect information events and explain them. Chapter 2 discussed artificial markets, sports betting markets and electronic markets for political events and other types of events. Those same events will now be used to test the method to see if a reasonable set of keywords or phrases can be extracted to explain them from the real world news stories [102].

This chapter concentrates on the fixed window approach of the keyword-based model and presents results from various markets including two markets from IEM (Iowa Electronic Market),¹ and one from the Foresight Exchange,¹ a market game that operates like a betting market (much like IEM), except that all transactions

¹<http://www.ideosphere.com/>

are using play money. Previous studies show that play-money market games behave in many ways like real markets [104, 105].

7.1 Event Description

7.1.1 Market Events

Market events are basically major and/or drastic price changes. While the real market or stock exchange data consists of price changes in several companies and will be described in the next chapter, for artificial markets, the following three markets are considered: (1) candidate Giuliani in the 2000 US New York Senate election, (2) candidate Gore in the 2000 US Presidential election, and (3) the outcome “extraterrestrial life discovered by 2050” (XLif) as defined on the Foresight Exchange². Note that in market (2), the winning bet was defined as the candidate with the largest share of the *popular* vote, not the winner of the electoral college, so Gore was the eventual winning bet.

7.1.2 Method and Notation

7.1.2.1 Logarithmic Score

The logarithmic score was used previously to measure accuracy and information incorporation in IEM. The logarithmic score is a *proper scoring rule* [132], and is an accepted method of evaluating probability assessments. When experts are rewarded according to a proper score, they can maximize their expected return by reporting their probabilities truthfully. Additionally, more accurate experts can expect to earn a higher average score than less competent experts. Suppose an expert reports probabilities $p_1, p_2, \dots, p_k$ for k mutually exclusive and exhaustive alternatives. Let $w_i = 1$ if and only if the i th event occurs, and $w_i = 0$ if it does not. Then the expert’s score for the current event is $\text{Log}(\sum_{i=1}^k w_i p_i)$. Higher scores indicate more accurate forecasts, with 0 the maximum and negative infinity the minimum. The “expert assessments” given by the market are taken to be the (normalized) prices of the candidates.

²<http://www.ideosphere.com/fx-bin/Claim?claim=XLif>

Note that under the logarithmic scoring rule, an expert's expected score equals the entropy of his or her probability distribution. Stated another way, the negative of the logarithmic score gives the amount that the expert is surprised by the actual outcome. So the logarithmic score applied to IEM is both a measure of forecast accuracy and a measure of the amount that the market is "surprised" when the winners of the elections are finally determined.

7.1.2.2 Notation

Let us assume that the probability of event E is $\mathcal{P}(E)$. Then the *likelihood* of an event E is

$$\mathcal{L}(E) = \frac{\mathcal{P}(E)}{\mathcal{P}(\bar{E})} \quad (7.1)$$

and the *log-likelihood* of E is

$$\mathcal{LL}(E) = \text{Log}\mathcal{L}(E) = \text{Log}\left(\frac{\mathcal{P}(E)}{\mathcal{P}(\bar{E})}\right) \quad (7.2)$$

We denote a gamble/bid paying off \$1 if and only if event E occurs as $\langle E \rangle$. Now let the price of $\langle E \rangle$ at time t be p_t . In analogy to the definitions of likelihood and log-likelihood, the *likelihood price* can be defined as

$$\mathcal{L}_t = \frac{p_t}{(1 - p_t)} \quad (7.3)$$

and the *log-likelihood price* as

$$\mathcal{LL}_t = \text{Log}(\mathcal{L}_t) = \text{Log}\left[\frac{p_t}{(1 - p_t)}\right]. \quad (7.4)$$

So, for example, given that the log-likelihood price is a at time $t - 1$, the probability that the log-likelihood price moves to b at time t is denoted by

$$\mathcal{P}(\mathcal{LL}_t = b | \mathcal{LL}_{t-1} = a). \quad (7.5)$$

Similarly, the probability that event E occurs given that the likelihood price at time t is a is written as $\mathcal{P}(E | \mathcal{L}_t = a)$.

7.2 Implementation

7.2.1 Finding out Pivotal dates

The price dynamics for the above markets were collected from IEM and foresight exchange price data. Daily price fluctuations in the above three markets were characterized using the difference between log-likelihood prices from one day to the next. That means that the log-likelihood price of day d_i is $\mathcal{LL}_{d_i}$ and that of the next day is $\mathcal{LL}_{d_{i+1}}$. Days were identified on which exceptionally large differences were observed. These dates are determined as follows. Assume the total number of days is D . For a day d_i the change in log-likelihood price $\Delta_{i,i-1}$ is

$$\Delta_{i,i-1} = \mathcal{LL}_{d_i} - \mathcal{LL}_{d_{i-1}} \quad (7.6)$$

We took the only dates d_j s for which $\Delta_{i,i-1} \geq \delta$, a threshold fixed by looking at the possible values.

7.2.2 Dates Found

Accordingly, the following dates were found to be some of the pivotal dates or “events”, immediately following huge price swings:

1. April 27, 2000 and May 19, 2000 for candidate Giuliani in the New York Senate market;
2. November 8, 2000 for candidate Gore in the US Presidential market;
3. August 6, 1996 for XLif on the Foresight Exchange.

Figures 7.2, 7.3, and 7.4 show the price graphs for the three markets surrounding these dates. Each of these dates was used as a splitting point for generating two text corpuses: a *negative* set of documents from before the date in question, and a *positive* set of documents from the week following the date in question (shown in Figure 7.1).

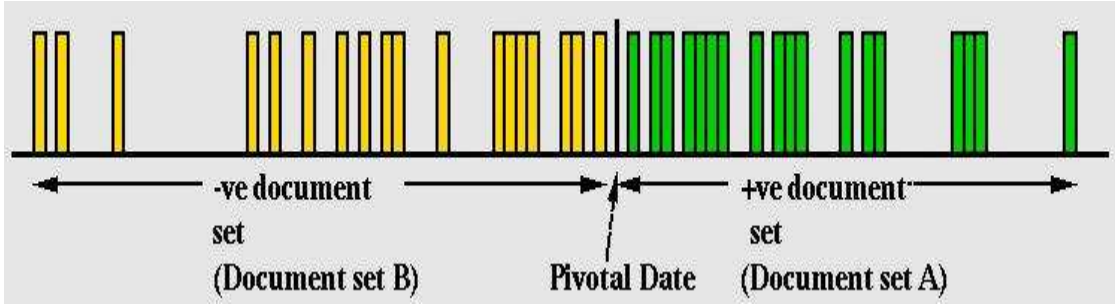


Figure 7.1. A simple way to show the analysis procedure. The documents are shown as vertical bars of different shades. The documents before the pivotal date (for which dates event we are looking for a reason) are in negative document set or Document B set. The documents after the pivotal date are put in the positive document set or Document A set.

7.2.3 Data

Documents were gathered from Usenet news archives on Google ³ for all three markets. We collected all postings to the newsgroups ny.politics (622 postings), us.politics (480 postings), and sci.space.news (127 postings) for the three markets, respectively. We did not use any keywords to narrow the search further. Additionally, we gathered the titles and abstracts of articles in the Washington Post containing “Giuliani” for the NY Senate market (189 articles). We then find the base time-line **BT** of these articles and sort them according to their base time-lines.

7.2.4 Our method

We identify the features (words and up to three-word phrases) that differentiate the positive and negative document sets using expected entropy loss. We do not explicitly remove stop words. Instead, we remove all features that occur in less than 7.5% of the positive documents. After they are sorted according to the base time-line or **BT** which is the publishing date and/or time there was no need of the phrases containing any date or time. So we remove all dates and numbers, and we also manually removed all source-specific words (e.g., “google” and “Washington Post”). We then rank keywords by expected entropy loss as follows.

³<http://groups.google.com/>

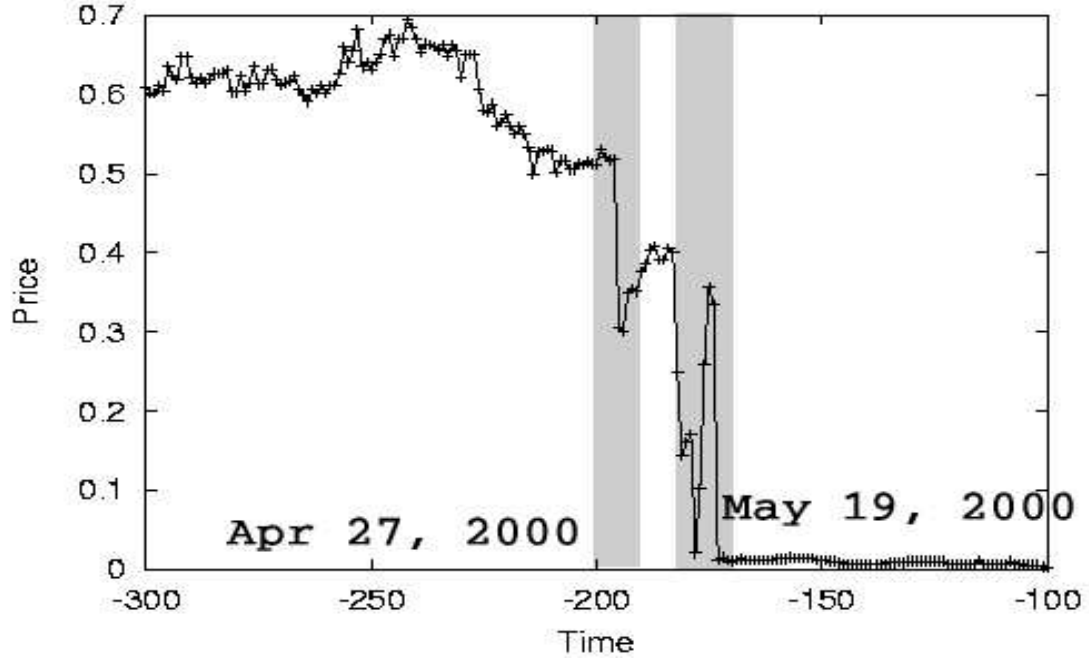


Figure 7.2. Portion of the price time series for candidate Giuliani in NY Senate election market on IEM, with significant dates highlighted.

7.2.4.1 Feature Entropy

Entropy is computed independently for each feature. Let P be the event that a document is in the positive set. Let f denote the event that the document contains the specified feature (e.g., contains the word “meteorite”). The prior entropy of the class distribution is

$$e = -Pr(P)LogPr(P) - Pr(\bar{P})LogPr(\bar{P}) \quad (7.7)$$

The posterior entropy of the class when the feature is present is

$$e_f = -Pr(P|f)LogPr(P|f) - Pr(\bar{P}|f)LogPr(\bar{P}|f) \quad (7.8)$$

likewise, the posterior entropy of the class when the feature is absent is

$$e_{\bar{f}} = -Pr(P|\bar{f})LogPr(P|\bar{f}) - Pr(\bar{P}|\bar{f})LogPr(\bar{P}|\bar{f}) \quad (7.9)$$

Thus, the expected posterior entropy is

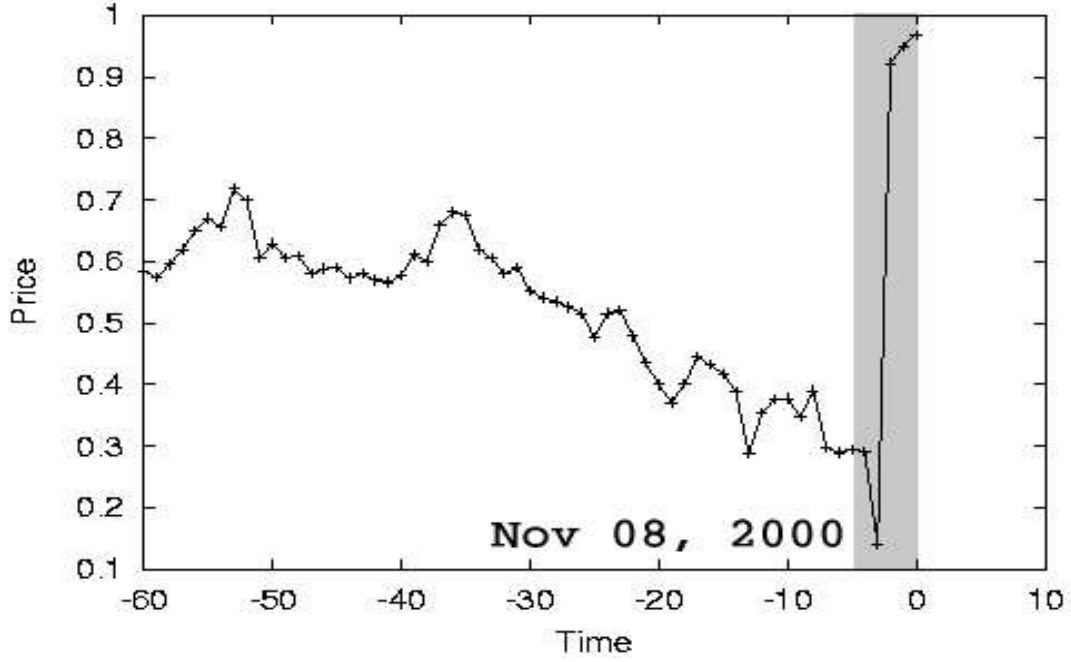


Figure 7.3. Portion of the price time series for candidate Gore in US Presidential election market on IEM. Note that in this market a bet for Gore won if Gore had a larger share of the *popular* vote.

$$e_f Pr(f) + e_{\bar{f}} Pr(\bar{f}), \quad (7.10)$$

and the *expected entropy loss* is

$$e - (e_f Pr(f) + e_{\bar{f}} Pr(\bar{f})). \quad (7.11)$$

If any of these probabilities are zero, we use a fixed value instead of 0 in above four equations. Expected entropy loss is synonymous with expected information gain, and is *always* nonnegative. Features are sorted by expected entropy loss to provide an approximation of the usefulness of each individual feature. This approach correctly assigns low scores to features that, although common in both sets, are unlikely to be useful for a binary classifier.

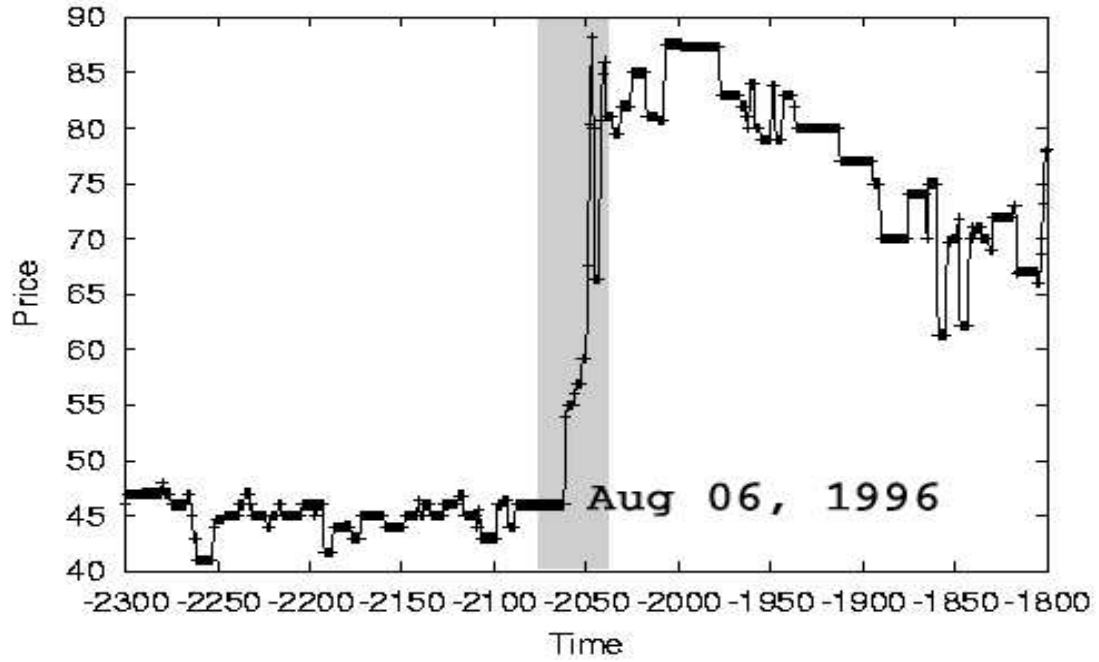


Figure 7.4. Portion of the price time series for “100 cents if extra-terrestrial life discovered by 2050” on the Foresight Exchange.

7.2.4.2 Kullback-Liebler Divergence

Kullback-Liebler divergence is another technique used here, by which we can get the probability distribution distance between two probabilities. As above, let P be the event that a document is in the positive set. Let f denote the event that the document contains the specified feature (e.g., contains the word “meteorite”). Then the Kullback-Liebler divergence for this word (and similarly for all other words) is

$$KL(f) = Pr(f|P) \text{Log} \frac{Pr(f|P)}{Pr(f|\overline{P})}. \quad (7.12)$$

The value of Kullback-Liebler divergence for any feature is always positive. We sorted the features (basically keywords/phrases) and took the top 10 list of keywords which indicates better probability of coming in the window A rather than window B.

7.3 Results

Results from analyzing the aforesaid (as described in subsection 7.1.1) three markets are shown in the Tables 7.1, 7.2, and 7.3. Our analysis could extract many words and phrases that are closely associated with real incidents. These incidents happened during the identified dates with major implications for the corresponding market bets.

1. NYSenate Election 2000 (NYSEnate)

For example, on April 27, 2000, Giuliani announced that he had prostate cancer. Entropy loss extracted terms and phrases like “cancer”, “prostate”, “prostate cancer” from both Usenet newsgroups and the Washington Post articles. Around May 19, 2000 Giuliani formally announced that he was quitting the Senate race, with Rick Lazio the replacement Republican candidate. Again, reasonable explanatory terms and phrases were discovered using our algorithm. Both the Usenet results (“lazio”, “rick lazio”, “mayor”, “voted”) and Washington Post results (“lazio”, “rick lazio”, “rick”, “rep rick”, “rep rick lazio”) indicate the name of Lazio as a top ranked feature. Words like “drop”, “dropped”, “quit”, and “bow out” did appear in the positive documents, but were removed during thresholding. We believe that more intelligent use of stemming and synonyms would help in this situation where there are many ways to say the same thing (as opposed to the case of “prostate cancer”, where there is essentially only one way to say it).

2. US Presidential Election 2000 (Press00)

In the US Presidential election, the price of candidate Gore skyrocketed after the election when it became clear he won the popular vote. The near tie and resulting chaos in counting the ballots in Florida appear in our extracted keyword list, where the top ranked features are “florida”, “ballots”, “recount”, “palm beach” etc.

3. Extra Terrestrial Life (XLif)

On August 6, 1996, NASA announced it had discovered possible signs of life on a Martian meteorite. The price of a bet on XLif on the Foresight Exchange rose quickly, apparently in response.

Source	Date	top ranked features
Usenet ny.politics	Apr 27, 2000	cancer, cancer newsgroups, prostate, prostate cancer, cancer newsgroups ny, commies subject liberal, subject liberal propaganda, damnliberals liberals, has prostate, commies
Washington Post	Apr 27, 2000	cancer, from prostate, is suffering from, business of politics, diagnosis, suffering from prostate, prostate cancer, suffering, from prostate cancer, cancer diagnosis
Usenet ny.politics	May 19, 2000	lazio, rick lazio, mayor, voted, rick, rep rick lazio, alt fan rush, difference, ca politics, families
Washington Post	May 19, 2000	lazio, rick lazio, rick, rep rick, rep rick lazio, convinced, opponent, rudy, abortion rights, giuliani's inner

Table 7.1. Top features found for dates corresponding to major price changes in the NY Senate market.

Source	Date	top ranked features
Usenet us.politics	Nov 08, 2000	florida, ballots, recount, palm beach, ballot, beach county, palm beach county, recounts, counted, county, in palm, the ballot, counties, fraud, in palm beach

Table 7.2. Top features found for the US Presidential market.

Indeed our algorithm found very relevant explanatory features, as listed in Table 7.3.

Source	Date	top ranked features
Usenet sci.space.news	Aug 06, 1996	meteorite, life, evidence, washington dc august, martian meteorite, primitive, gibbon, organic, of possible, martian, life on mars, david, life on, billion years ago, mckay

Table 7.3. Top features found for the XLif market on the Foresight Exchange.

7.4 Conclusion

In this chapter we describe our first basic model of explanatory keyword extraction based on entropy model as well as equivalent Kullback-Liebler divergence model. In three case studies, our algorithm found keywords and phrases subjectively very relevant to the events corresponding to sharp market upswings or downswings or directly or indirectly responsible for these upswings and downswings. We believe that this model can also be used for the real market and we show the experimental results in the next chapter.

Chapter 8

Keyword-based Model and Need of an Ontology

*“Knowledge is of two kinds. We know a subject ourselves, or we know
where we can find information on it.”*

- Samuel Johnson

(English author, critic, and lexicographer)

(1709 - 1784)

8.1 Prelude

As we see from the last chapter, basic keyword-model works well in artificial markets with artificial money and artificial market with real money. Now let us concentrate on the real market with real money – which is the stock market. We considered here around 200 companies enlisted in the Forbes list of global companies [13]. This list gives us a look at the top global companies and we took top 200 US companies from that list as it is easy to follow up the company profiles, news articles (regarding language), and timelines (other countries may have a different way of representing the timeline) without any major change in our system.

8.2 Real Market Analysis

Real market with real money, at first look, seems to be no different than artificial market regarding the basic characteristics. For example the artificial markets also show the characteristics of efficient market hypothesis [103] and we will give excerpt from his paper here:

There are various forms of the efficient markets hypothesis, and at least four different degrees of efficiency to consider:

1. *Internal coherence: prices are self-consistent or arbitrage-free: no trader can make a sure profit without any risk.*
2. *Internal unpredictability: future prices are not predictable based on current and past prices. Also called the weak form of the efficient markets hypothesis.*
3. *Unpredictability: future prices are not predictable based on any currently available information, including prices, economic variables, fundamental data, etc. Also called the semi-strong form of the efficient markets hypothesis.*
4. *Expert-level accuracy: Prices fully reflect all information available to all traders. Informed experts cannot consistently outperform naive traders. In particular, when prices constitute forecasts, market estimates are at least as accurate as expert assessments. Also called the strong form of the efficient markets hypothesis.*

He also mentions about the *Theory of rational Expectations* and that it hold also in the artificial market scenario.

While there is no doubt that there are lots of characteristic similarity between artificial and real market, however to generate the keyword explanation of any given event, things are not always the same. While our keyword generation algorithm solely depends on the news articles about the event and a stream of it (as described in chapter 6), the frequency and characteristics of these articles are much different from regular news articles.

8.2.1 Differences

Now the basic difference in the form of news articles and their focus and availability is as follows:

- Events in artificial markets such as IEM political markets or the Foresight exchanges long bets are associated with news articles of slow changing nature. Whereas the real stock market is associated with fast change of price and fast change of news articles and trends.

- News articles about artificial market events are more in number due to the above nature of long duration and for a single major event (less in number than the real stock market), it is easy to find more number of news articles.

In case of real stock exchange there are more number of events in a small time-frame and so the number of news articles, which can associated with a major event are much less than in the case of artificial market.

- News articles for the artificial market events are general news in nature, such as NASA's discovery of life evidence in Mars is a general science related news and all news channels cover this type of news. Therefore the possibility of getting a huge number of articles is more in this kind of news event.

On the contrary a useful news article of a company, for our analysis purpose is of specific nature. It is usually pre-filtered for the specific company and as a result can not be well distinguished from other news articles which are also pre-filtered in the same way but talk about very different events. It is hard to find deeper relevancy and distinguish them if we simply use the company name or ticker symbol as a keyword for TFIDF analysis. As the number of articles are also limited in financial domain, it becomes harder to analyze.

8.3 Evaluation of Keyword-based Model

From the previous section we get an idea of difference between artificial market explanation generation and real stock market explanation generation, but do not get any evidence. In this section we show with some evidence that the bare entropy

model is not sufficient in the later case. In tables 8.2, 8.3, 8.4, 8.5, and 8.6, we show the total dataset with number of upswings and downswings of prices for 200 companies from Forbes list of US companies. These price swings are also aligned with the news articles about them crawled from various websites through Yahoo finance page¹. The number of articles associated with these swings are also given in the fifth and sixth column and thus the total number of articles in the seventh column. These dataset represents the price dynamics of just over an year (July 2004 - Aug 2005) and the price swings are registered only if the total percentage difference is more than 2%.

The results from these many events will be huge in size and so we prefer to show them on the graph for 20 different companies and we show the scoring of these explanations in figure 8.2.

8.3.1 Metric

Results from any explanation model is hard to quantify. As far as our knowledge in this area of research goes, there is no prior effort in explaining market events, and to the best of our knowledge we are the first to try it. Therefore it is not a completely established field of research and there are no fixed data-archives (like TREC), no benchmark algorithms, no well-defined metric or any chance of a performance comparison. We try to come up with an acceptable metric by which we can at least evaluate our system with or without different components (such as window type² or ontology³ etc.).

8.3.1.1 Weighted Precision

To create the metric useful for quantifying the performance, we followed the IR research carefully and came up with “weighted precision” metric. Here we demonstrate our thought process.

- Precision in IR community is defined as the ratio between the number of relevant retrieved item and the total retrieved item. In short, precision is

¹<http://finance.yahoo.com>

²Fixed-windows vs. Variable-window

³With or without using ontology

basically “how much of the retrieved item set is relevant”. In mathematical notation if from any engine (let us suppose search engine or an explanation generation engine), if the result contains t item (documents or keywords) and out of these items, only r items are relevant to the query (search engine query or price-swing query), then the precision Pr of the result is

$$Pr = \frac{r}{t} \quad (8.1)$$

- We also know about the recall metric which is nothing but “how much of the relevant items are retrieved”. It means that if we somehow know beforehand that how many relevant items the aforesaid engine would provide in the ideal case, then we can quantify the recall performance by the ratio of the number of relevant items retrieved and the total number of relevant items in the whole universe in question. Therefore if the number of relevant items retrieved is r (as in the previous equation) and the total number of relevant items in the universe, which has been missed by the engine is m , then the total number of relevant items in the universe is $r + m$ and the recall Re is

$$Re = \frac{r}{r + m} \quad (8.2)$$

- Now comes the difficult part. First of all we do not know for sure how many different factors played a role behind the price swing. The factors playing behind any price change can be almost impossible to fathom completely. Sometimes there are generic factors (not specific to the company) and in this work *we do not consider the generic effect*. So finding out the recall value is really difficult as we do not know the amount m in the equation above.
- So now we have to rely on precision. We can at least find out how many of the extracted keyword can be associated with a fact or possible reason behind any price movement. Usually this is not too hard to find. Nowadays financial websites are full of analysts’ columns and we can at least find out for sure what major things happened or any breaking news. *Therefore, calculating precision will not be too hard..*

- Now we have to think whether the precision itself will be enough for quantifying the betterness of a solution over another. In our belief, mere precision can not distinguish between a result having 4 useful keywords in the end of a top 10 keyword versus a result having the same 4 words in the beginning of the list. Simply, we will not be able to quantify the ranks of the resulting keywords if we just consider the plain vanilla precision.
- **Solution:** We came up with “weighted precision” measure. This is basically giving more weight to the keywords according to their ranks in the result list.

If we denote weighted precision as Pr_w , then

$$Pr_w = \frac{r_w}{t_w} \quad (8.3)$$

where

$$r_w = \sum_i^r S_i \quad (8.4)$$

and S_i is the scoring which

$$t_w = \max(\sum_i^r S_i) \quad (8.5)$$

and so,

$$Pr_w = \frac{\sum_i^r S_i}{\max(\sum_i^r S_i)} \quad (8.6)$$

For all our experiments, we took 10 top keywords in the result. Now we can assign different weight to them according to their position in the list. So the topmost keyword (if it is considered as a keyword which can be associated with an explaining event) will get $10X$ point. The next keyword (again if it is considered as a keyword which can be associated with an explaining event) will get $9X$ point and so on. As the maximum point a list can get is

$$\max X(1..10) = \sum_i^{10} (i \times X) = 55X \quad (8.7)$$

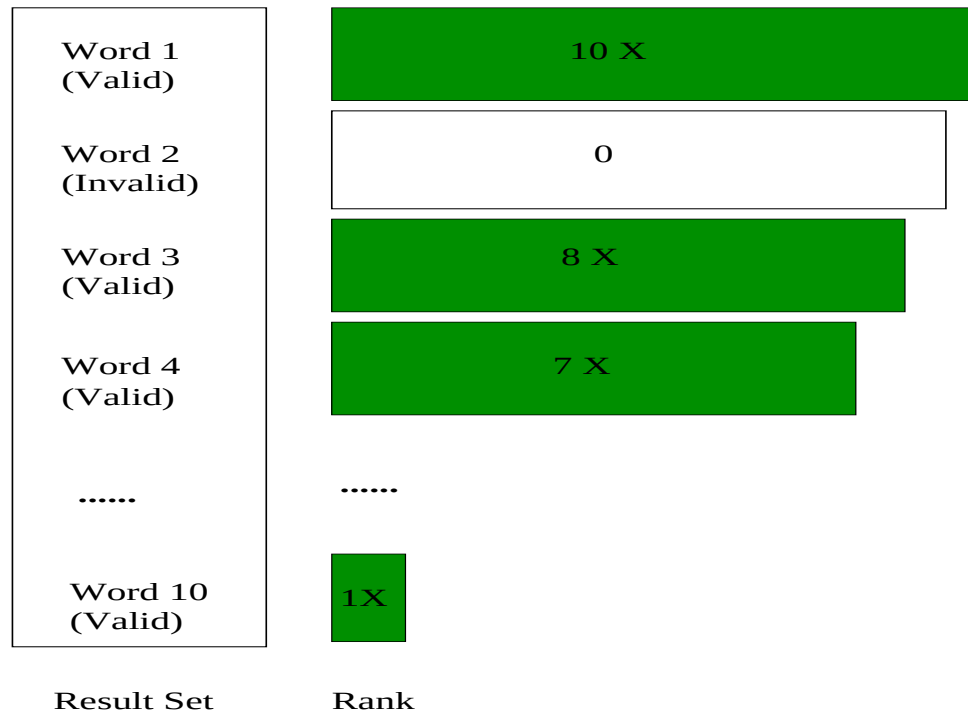


Figure 8.1. The ranking of an explanatory keyword-set. The top ranked keywords, which can be associated with an explanatory event will be given highest score. The total score is divided by the maximum possible score of $55X$ (for top 10 keywords) for normalization purpose.

therefore we normalize the all the ranking by dividing them with $55X$. This procedure is shown in figure 8.1. This is a way to ensure that we give high weight to a keyword which is an explanatory keyword and also coming in a top rank.

- This method is a restrictive method as the overall ratio only tells us how many percentage of the resulting keyword-set can be considered as a set where all the keywords are explanatory. So scoring a good point is hard. However this scoring technique has the feature that a top scoring single keyword at first position having $10X$ points is considered to be same as three keywords at

positions 6th, 7th and 9th position (having scored $5X + 4X + 1X = 10X$). Our argument is that both the rank and the number of keywords should be counted. So it is a good mix of both.

8.3.2 Results

Now we are ready to show the preliminary results from the basic keyword model with windows of fixed width. Though this same technique gave us promising results in the artificial market (as seen from chapter 7) it does not give very good results. We evaluated 20 major companies (as shown in table 8.1) and we took top 10 price swings for each one of them to calculate the average “weighted precision” score. The scores are shown in figure 8.2.

So as we see from the results graph, the entropy model with fixed window approach does not work well. Therefore we need to improve the model. We plan to do so by (1) the variable window approach and (2) using better relevance model. In the next few sections we describe the need for the better relevance model, a proof-of-concept system design and then how we can use this relevance model to improve the results.

8.4 Need of a Better Relevance

In this section we present one of our previous work on an intelligent agent system, where domain knowledge has been successfully used before. The system is called **LawBOT** and it is described in detail in our IEEE Intelligent Systems paper [52]. Here we give a brief description of this system and its architecture and show that the same ideas can be fruitfully used in case of business domain.

8.4.1 An Agent Based Solution

In this section we present an Internet based agent which was designed to assist legal researchers in retrieving laws and case reports electronically warehoused at a diverse set of databases maintained by local, state, and federal governments.

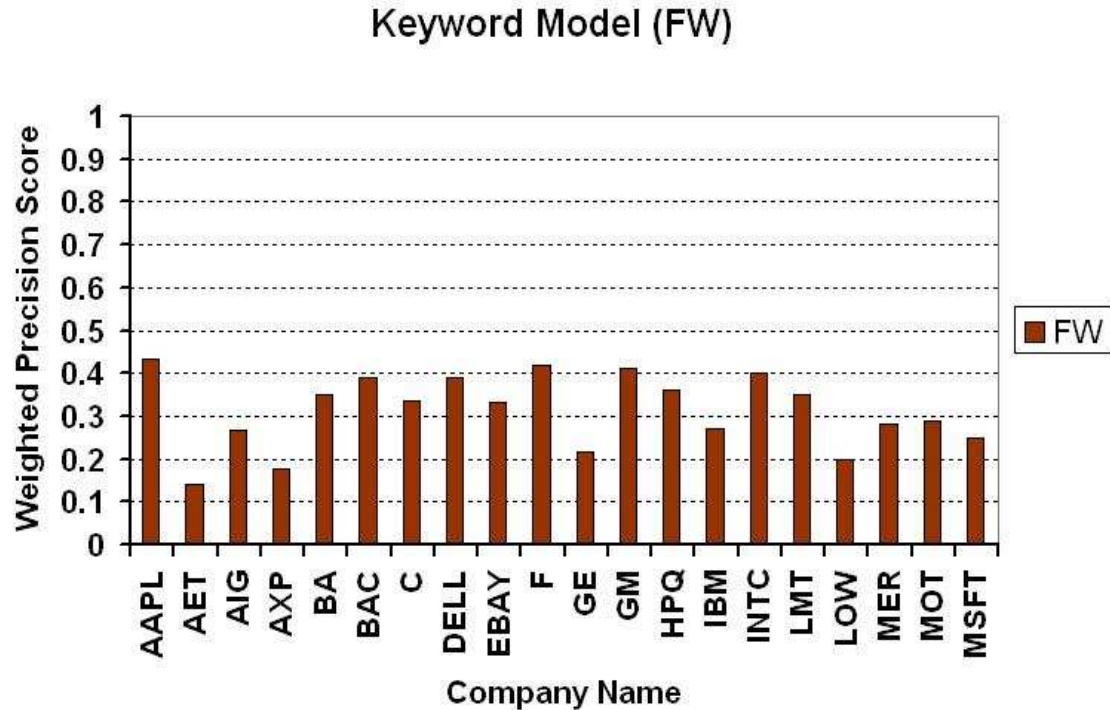


Figure 8.2. The results from first 10 companies. The score is pretty low for plain vanilla “fixed window” approach.

8.4.1.1 LawBOT: A Multiagent Assistant for Legal Research

LawBOT is implemented as a collection of agents which are employed according to users preferences to collect, filter, organize and recommend relevant case histories, state statutes or supreme court cases. Our goal is to create a system that can be effectively used not only by lawyers but also by a lay person to retrieve legal documents relevant to the issue that the user wants to research. The requirement of enabling research by the commoner required us to add a novel ontology-based search component. We have developed an ontology for some of the common law categories and will show that similar techniques can also be applied to business domain. This ontology is used to map colloquial terms to corresponding legal

Ticker	Name	UP Count	DOWN Count	UP Doc	Down Doc	Total Document to Analyze
AAPL	Apple Computer	43	33	439	240	679
AET	Aetna	32	18	45	44	89
AIG	American Intl Group	23	24	143	140	283
AXP	American Express	13	12	66	29	95
BA	Boeing	30	19	134	136	270
BAC	Bank of America	12	14	134	273	407
C	Citigroup	13	10	127	201	328
DELL	Dell	21	16	278	110	388
EBAY	eBay Inc.	41	31	184	214	398
F	Ford Motor	27	30	209	255	464
GE	General Electric	16	10	44	17	61
GM	General Motors	28	26	515	481	996
HPQ	Hewlett-Packard	26	24	300	164	464
IBM	IBM	18	14	565	419	984
INTC	Intel	33	29	501	430	931
LMT	Lockheed Martin	23	17	123	58	181
LOW	Lowe's Cos	26	20	101	28	129
MER	Merrill Lynch	23	18	163	60	223
MOT	Motorola	37	25	342	75	417
MSFT	Microsoft	14	13	534	129	663

Table 8.1. Details of the dataset for the randomly selected 20 companies. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model.

terminology. This feature enables the average user to perform a more effective and thorough search for relevant legal documents. The ontology also enables query enhancement to search by related words which can return a more comprehensive set of documents. In business domain, we will show a set of techniques which can be used to do the similar job efficiently.

8.4.1.2 LawBOT Architecture

The system architecture is presented in Figure 8.3. In the following we briefly state the functionalities of each module:

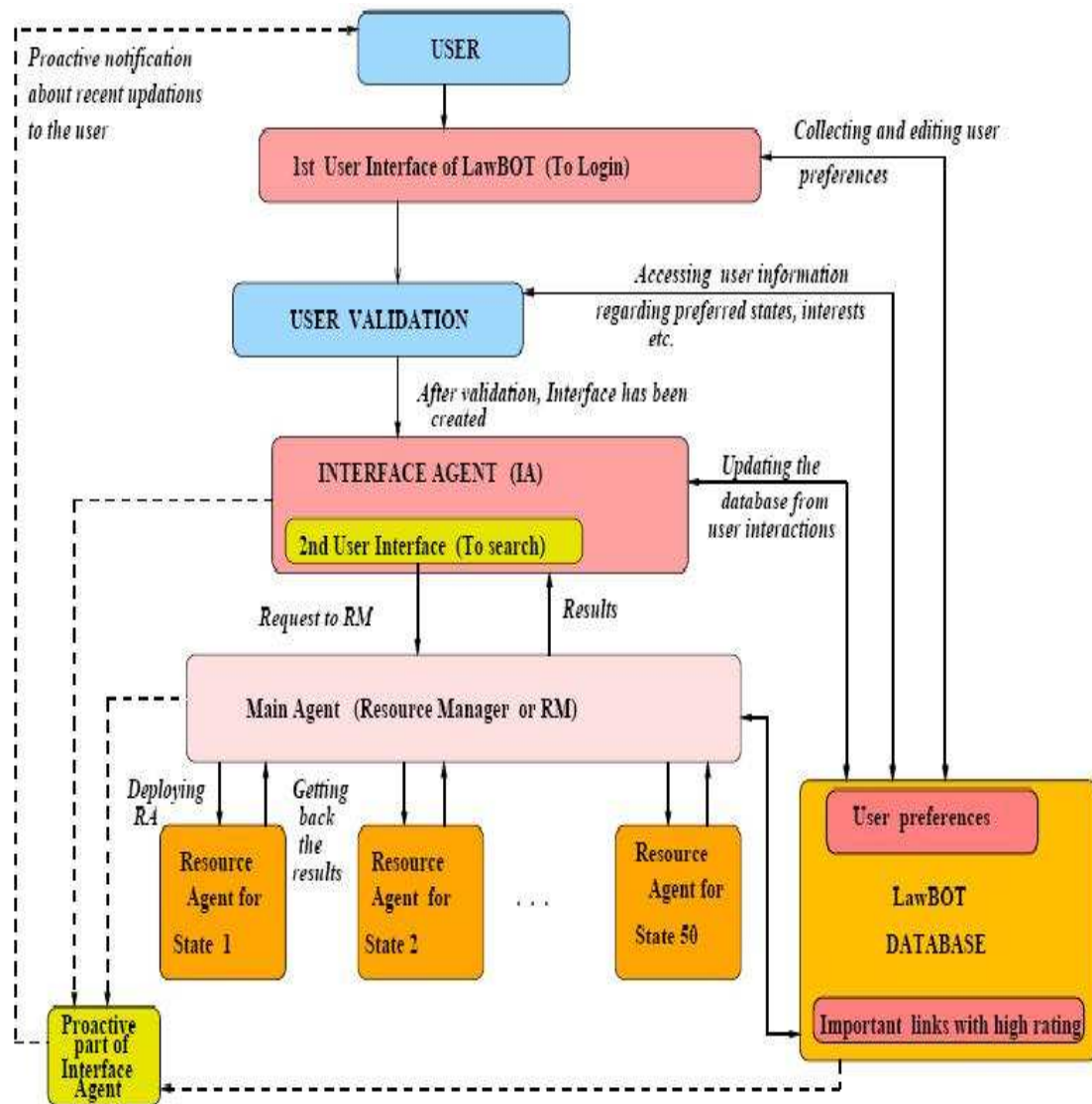


Figure 8.3. The architecture of the LawBOT system designed to use ontologies in the context of a legal database search.

- **Interface:** The first screen is used to login to the system. User profiles including preferences and recent searches are stored in the system. These are used to present a customized second screen (e.g., allowing users to resubmit modified versions of recent searches, biasing search to look for specific states, etc.). The interface connects to the user preference database to collect or edit the user interactions and preferences. The interface forwards user queries to

resource managers (described next) and also displays to the user the results provided by the resource manager.

- **Resource Manager:** Resource Manager (RM) plays the important role of organizing information retrieval and processing. It augments the user query by consulting domain ontologies and then deploys the relevant Resource Agents as required by the query.

The RM fuses the information retrieved by resource agents and from existing LawBOT databases and returns this to the interface. The RM can also perform proactive notification to the users with recent updates on prior search results if requested. This feature allows the user to be updated with important and timely information as new laws and cases become available.

- **Resource Agent:** The user's selection of jurisdictions to be searched determines which resource agent (RA) or agents will receive search requests. On receiving requests, the research agents format them according to the syntax expected by the search engine of the jurisdiction to be searched. The RAs are programmed to access specific sites on the web and run a word filtering algorithm over the retrieved documents. The documents are sorted and ordered according to word densities. The ordered documents are returned to the Resource Manager using an Agent Communication Language (ACL) [9, 11].
- **LawBOT Database:** The LawBOT database stores user preferences, domain ontology, and auxiliary information like frequently used links, etc. which the RM can then effectively use to reformulate queries and rank results.
- **Proactive Component:** The proactive component can use stored user preferences and past queries to do more extensive research offline, i.e., when the user is not logged in. If new documents with high relevance are found in this offline search, the user will be notified about the availability of new documents relevant to their interest.

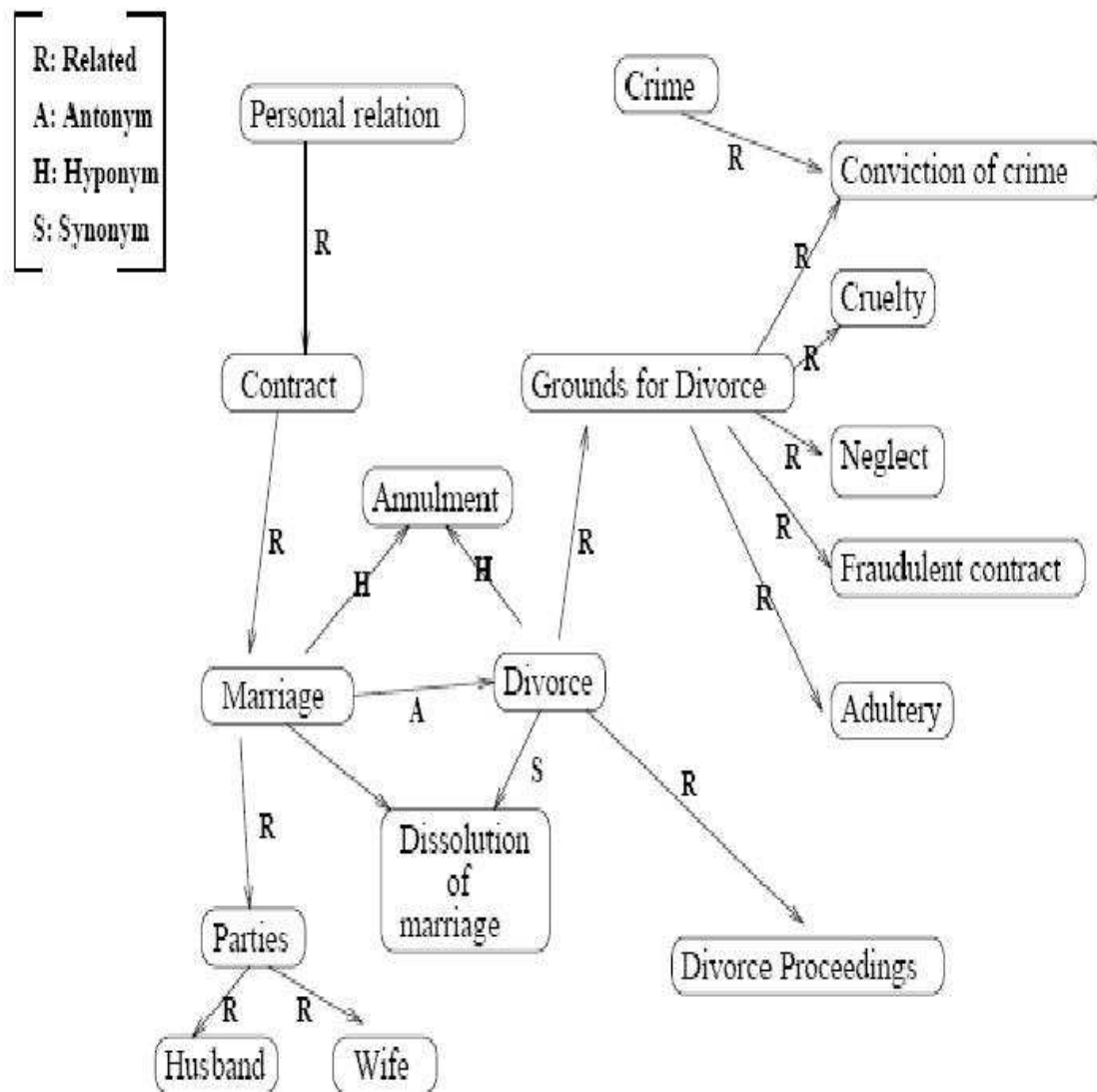


Figure 8.4. A part of the ontology used by the LawBOT system. The ontology here is shown as a connected graph of related keywords.

8.4.2 Using Ontology to Augment Search

As one of the principal goals for developing LawBOT was to enable the lay person to retrieve legal documents of interest, additional functionality had to be provided to rephrase an informal query into legal jargon. The use of an extensive ontology [131] was planned to enable this functionality. Ontology helps to build proper relations between words and phrases and those relations can be used to reformulate

a query if very few or no results are available. It is very difficult for a common person to find the documents efficiently and quickly due to the lack of knowledge of legal terminology. Ontology allows the user to reformulate a particular phrase into proper, closely related legal jargon. The facility to reformulate a query helps naive users to find legally useful documents from a very large database. Some examples are provided which explains the kind of functionality we planned for and then the outline of our approach to obtaining such a capability is presented.

- **Legally Appropriate Synonyms:** The word kid is similar in meaning to baby, child, or youngster. Of these, child is used most frequently in law records. So any search with the words kid or baby will rarely produce any worthwhile result. LawBOT uses its legal ontology to search by child as well when a naive user chooses kid as the search word.
- **Related Words and Law Categories:** The word child may be associated with a number of other words, e.g., care, custody, support, abuse, etc. A user who searches only with child will receive too many references. This system assists the user to rephrase such searches by asking them to select either from a law category (e.g., family law, criminal law), or by expanding the search by choosing from a suggested list of often-used additional terms.
- **Appropriate Search:** Legal issues are categorized into standardized, fixed sections. The number is indicative of the category of the law. A search by 26 or Section 26 should initiate the search for Internal revenue related laws. This is unique in the context of legal research. Such information is rarely used in the search engines which use either word count or word density. Rather, common search engines often returns irrelevant pages which included 26 in the date (April 26, 1996), or as in Local 26 or in 26th Delaware Street.

This ontology is a semantic network (figure 8.4) which relates words by their relationships. Words are associated with synonyms, hyponyms, hypernyms (this, for example, allows the system to additionally search for firearm laws if someone asks for handgun laws), often-used associated words, special synonyms (e.g., child for baby), etc. If the user-specified keywords contain a special synonym, the latter is used in place of the given keyword. If the number of documents returned

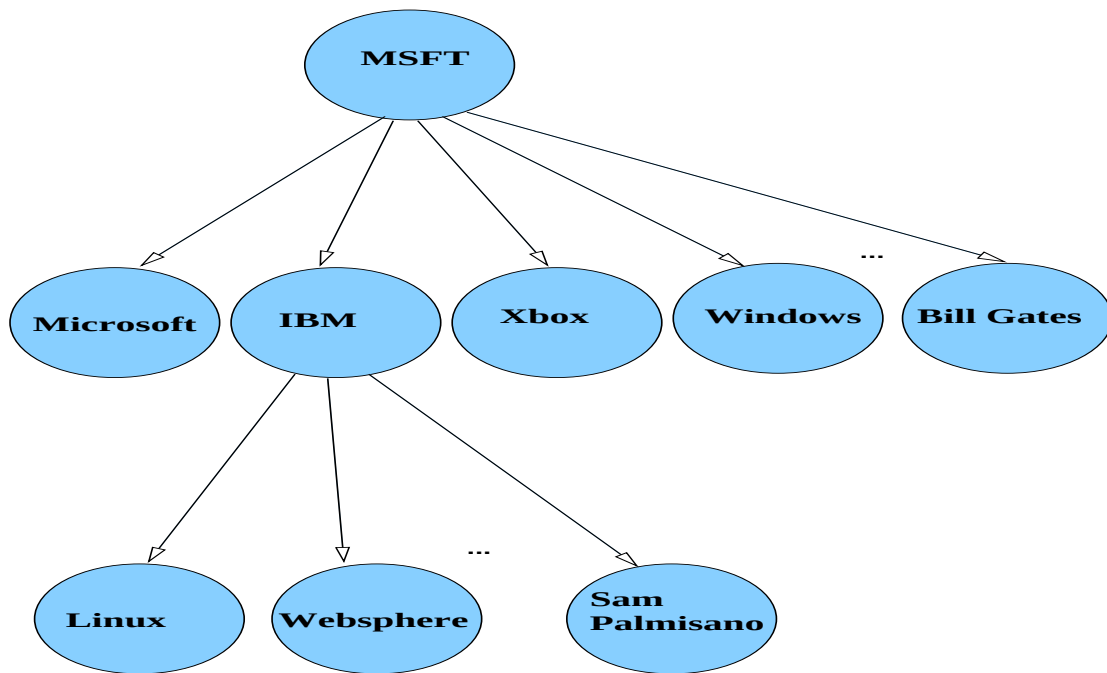


Figure 8.5. A proposed ontology for Microsoft

for the user-specified search is below a threshold, the search is augmented by using synonyms, hyponyms, and hypernyms. On the other hand, if the user-specified search returns too many results, the user is asked to choose from law categories and/or augment search with often-used associated words. The current system uses a core ontology with terms from family law. This was appropriate to demonstrate the proof-of-concept of ontology-based search augmentation, but needs to be expanded to other law categories.

8.4.2.1 A Proposed Business Ontology

It is hoped that the use of ontology like in LawBOT can generate better results. The news articles crawled are only superficially filtered according to the company ticker symbol, and this can be improved. Moreover, instead of taking the whole article for explanatory keyword generation, it can be narrowed it down to only those few paragraphs or sentences which are absolutely important and relevant

to the specific company. Explanation generation is a complex procedure and any noise filtered out can improve the overall performance of the system.

In this regard this study foresees building business ontologies. These are ontologies for individual companies that work as a knowledge source for that company. It is hoped that they can be efficiently exploited to extract the most important portions from a large news article.

8.5 Our Goal

In the next chapter we show the intricate details of how we can build a business ontology and then in chapter 10 will show the results from the keyword based model with the use of these ontologies. For sentence based model, we also used the ontology as part of the whole system and it improved the system performance too.

Ticker	Name	UP Count	DOWN Count	UP File- count	Down File- count	File to Ana- lyze
AA	Alcoa	31	22	129	98	227
AAPL	Apple Computer	43	33	439	240	679
ABC	AmerisourceBergen	26	18	31	4	35
ABS	Albertsons	28	26	14	46	60
ABT	Abbott Laboratories	17	14	42	61	103
ADBE	Adobe	35	25	39	66	105
ADM	Archer Daniels	34	21	32	7	39
ADP	Automatic Data	19	14	22	10	32
AET	Aetna	32	18	45	44	89
AFL	Aflac	22	16	18	13	31
AHC	Amerada Hess	34	27	22	7	29
AIG	American Intl Group	23	24	143	140	283
ALL	Allstate	23	11	29	18	47
AMGN	Amgen	25	23	131	47	178
AOC	Aon	27	18	50	14	64
APA	Apache	40	29	17	33	50
APC	Anadarko Petroleum	38	27	43	13	56
AT	Alltel	16	11	29	10	39
AXP	American Express	13	12	66	29	95
BA	Boeing	30	19	134	136	270
BAC	Bank of America	12	14	134	273	407
BAX	Baxter International	22	12	34	6	40
BBT	BB &T	17	14	33	14	47
BBY	Best Buy	39	25	133	42	175
BK	Bank of New York	18	18	28	10	38
BLS	BellSouth	16	17	120	50	170
BMJ	Bristol-Myers Squibb	21	19	67	79	146
BNI	Burlington Santa Fe	34	19	48	7	55
BR	Burlington Resources	43	34	20	5	25
BRK-A	Berkshire Hathaway	11	12	28	30	58
BSC	Bear Stearns Cos	23	20	86	32	118
BUD	Anheuser-Busch Cos	3	12	5	23	28
C	Citigroup	13	10	127	201	328
CAG	ConAgra Foods	14	18	11	39	50
CAH	Cardinal Health	25	23	39	32	71
CAT	Caterpillar	29	21	116	21	137
CB	Chubb	25	12	26	4	30
CCE	Coca-Cola Enterprises	22	20	32	11	43
CCU	Clear Channel Commun	21	24	48	60	108
CD	Cendant	23	20	36	64	100

Table 8.2. Details of the dataset. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model.

Ticker	Name	UP Count	DOWN Count	UP File- count	Down File- count	File to Ana- lyze
CEG	Constellation Energy	24	15	26	9	35
CFC	Countrywide Financial	28	26	29	33	62
CI	Cigna	31	18	62	11	73
CIT	CIT Group	27	21	35	23	58
CL	Colgate-Palmolive	16	15	51	40	91
CMA	Comerica	21	17	30	10	40
CMCSA	Comcast	22	22	115	135	250
COF	Capital One Financial	23	17	28	38	66
COP	ConocoPhillips	38	26	162	80	242
COST	Costco Wholesale	28	24	34	30	64
CSC	Computer Sciences	29	21	32	22	54
CSCO	Cisco Systems	28	31	380	206	586
CTX	Centex	39	36	27	19	46
CVS	CVS	28	15	40	9	49
CVX	ChevronTexaco	31	20	267	197	464
DD	EI du Pont de Nemours	21	17	39	78	117
DE	Deere & Co	30	25	44	19	63
DELL	Dell	21	16	278	110	388
DIS	Walt Disney	22	15	151	53	204
DOM	Dominion Resources	27	18	3	0	3
DOW	Dow Chemical	28	21	65	28	93
DTE	DTE Energy	18	11	13	16	29
DVN	Devon Energy	38	34	21	18	39
EBAY	eBay Inc.	41	31	184	214	398
ED	Consolidated Edison	16	5	23	0	23
EIX	Edison International	26	10	12	12	24
EK	Eastman Kodak	28	21	57	82	139
EMC	EMC	38	29	61	66	127
EMR	Emerson Electric	21	12	30	0	30
ETR	Entergy	20	13	30	20	50
EXC	Exelon	31	19	95	26	121
F	Ford Motor	27	30	209	255	464
FD	Federated Dept Strs	32	24	59	28	87
FDC	First Data	25	25	23	31	54
FDX	FedEx	30	21	77	78	155
FE	FirstEnergy	25	10	26	7	33
FITB	Fifth Third Bancorp	18	24	11	46	57
FNM	Fannie Mae	22	27	91	113	204
FON	Sprint FON	29	16	184	18	202
FPL	FPL Group	22	8	26	3	29
FRE	Freddie Mac	28	20	78	52	130

Table 8.3. Details of the dataset. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model.

Ticker	Name	UP Count	DOWN Count	UP File- count	Down File- count	File to Ana- lyze
G	Gillette	20	16	60	27	87
GCI	Gannett	10	11	23	79	102
GD	General Dynamics	18	12	45	13	58
GDW	Golden West Finl	26	20	16	1	17
GE	General Electric	16	10	44	17	61
GIS	General Mills	11	10	6	48	54
GM	General Motors	28	26	515	481	996
GP	Georgia-Pacific	27	24	25	21	46
GPS	Gap	29	28	66	50	116
GS	Goldman Sachs Group	20	13	204	90	294
HCA	HCA	23	21	42	33	75
HD	Home Depot	25	21	148	59	207
HNZ	HJ Heinz	14	13	22	22	44
HON	Honeywell	28	23	53	45	98
HPQ	Hewlett-Packard	26	24	300	164	464
IBM	IBM	18	14	565	419	984
INTC	Intel	33	29	501	430	931
IP	International Paper	24	28	53	34	87
ITW	Illinois Tool Works	17	16	9	1	10
JCI	Johnson Controls	21	19	12	15	27
JCP	JC Penney	40	26	70	27	97
JHFT	John Hancock Financial	20	11	0	0	0
JNJ	Johnson & Johnson	16	7	42	73	115
JPM	JP Morgan Chase	15	14	98	61	159
K	Kellogg	10	8	37	5	42
KEY	KeyCorp	22	13	14	11	25
KMB	Kimberly-Clark	12	12	6	26	32
KO	Coca-Cola	17	13	169	70	239
KR	Kroger	24	16	20	31	51
KRB	MBNA	24	20	25	77	102
KSS	Kohl's	33	28	36	34	70
LEH	Lehman Bros Holdings	29	19	99	34	133
LLY	Eli Lilly and Co	30	30	118	89	207
LMT	Lockheed Martin	23	17	123	58	181
LNC	Lincoln National	23	19	18	17	35
LOW	Lowe's Cos	26	20	101	28	129
MAS	Masco	30	19	12	9	21
MAY	May Dept Stores	28	25	42	11	53
MCD	McDonald's	25	20	101	95	196
MCK	McKesson	32	28	52	19	71
MDT	Medtronic	26	21	82	49	131

Table 8.4. Details of the dataset. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model.

Ticker	Name	UP Count	DOWN Count	UP File- count	Down File- count	File to Ana- lyze
MEL	Mellon Finl	20	19	9	10	19
MER	Merrill Lynch	23	18	163	60	223
MET	MetLife	27	18	72	26	98
MHS	Medco Health Solutions	34	27	25	14	39
MMC	Marsh & McLennan	26	21	41	95	136
MMM	3M	19	19	28	30	58
MO	Altria Group	21	18	58	58	116
MOT	Motorola	37	25	342	75	417
MRK	Merck & Co	26	23	233	259	492
MRO	Marathon Oil	41	26	52	29	81
MSFT	Microsoft	14	13	534	129	663
MTB	M&T Bank	24	14	15	8	23
MWD	Morgan Stanley	32	30	162	104	266
NCC	National City	20	17	29	15	44
NKE	NIKE	27	16	68	47	115
NOC	Northrop Grumman	16	19	87	70	157
NSC	Norfolk Southern	37	24	35	20	55
NXTL	Nextel Commun	30	28	117	34	151
OMC	Omnicom Group	18	21	31	37	68
ORCL	Oracle	32	30	130	91	221
OXY	Occidental Petroleum	34	23	38	30	68
PBG	Pepsi Bottling Group	20	19	23	13	36
PCAR	Paccar	36	25	11	12	23
PEG	Pub Svc Enterprise	26	17	40	8	48
PEP	PepsiCo	15	13	93	32	125
PFE	Pfizer	21	25	266	376	642
PFG	Principal Financial	26	17	19	2	21
PG	Procter and Gamble	18	16	110	70	180
PGN	Progress Energy	13	15	11	2	13
PGR	Progressive	24	15	10	19	29
PNC	PNC Financial Services	17	14	24	10	34
PPL	PPL	22	12	34	4	38
PRU	Prudential Financial	25	14	25	10	35
Q	Qwest Communications	38	35	138	140	278
QCOM	Qualcomm	39	36	121	58	179
RF	Regions Financial	16	11	12	10	22
RHAT	Red Hat	44	41	37	38	75
RTN	Raytheon	17	11	10	7	17
SBC	SBC Communications	16	13	195	123	318
SCH	Charles Schwab	38	29	95	25	120
SEBL	Siebel Systems Inc.	42	42	34	81	115
SGP	Schering-Plough	28	24	86	36	122

Table 8.5. Details of the dataset. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model.

Ticker	Name	UP Count	DOWN Count	UP File- count	Down File- count	File to Ana- lyze
SHLD	Sears, Roebuck	41	38	100	86	186
SLE	Sara Lee	20	16	15	40	55
SLM	SLM	28	21	24	22	46
SNE	Sony Corp.	22	18	85	128	213
SO	Southern Co	15	8	30	9	39
SPLS	Staples	33	33	26	11	37
SRE	Sempra Energy	25	13	47	27	74
STA	St Paul Cos	23	13	41	17	58
STI	SunTrust Banks	22	13	18	4	22
STT	State Street	15	16	15	34	49
SYX	Sysco	16	22	11	15	26
T	AT&T	24	19	123	67	190
TGT	Target	35	20	117	54	171
TRB	Tribune	18	21	30	57	87
TWX	Time Warner	22	23	152	271	423
TXN	Texas Instruments	40	35	221	77	298
UCL	Unocal	35	21	193	74	267
UNH	UnitedHealth Group	37	18	43	83	126
UNP	Union Pacific	22	18	31	14	45
UPS	United Parcel Service	18	17	42	69	111
USB	US Bancorp	15	15	21	16	37
UTX	United Technologies	24	14	25	27	52
VIA	Viacom	18	25	73	132	205
VLO	Valero Energy	49	34	74	51	125
VZ	Verizon Communications	13	15	105	122	227
WAG	Walgreen	24	19	65	11	76
WB	Wachovia	17	16	53	74	127
WFC	Wells Fargo	12	8	20	19	39
WLP	WellPoint Health	27	22	44	35	79
WM	Washington Mutual	20	15	39	42	81
WMI	Waste Management	19	17	9	16	25
WMT	Wal-Mart Stores	14	16	108	132	240
WY	Weyerhaeuser	27	18	35	21	56
WYE	Wyeth	25	14	44	24	68
XOM	ExxonMobil	26	19	195	197	392
XRX	Xerox	31	27	15	92	107

Table 8.6. Details of the dataset. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model.

Chapter 9

Domain Ontology Construction

“The only good is knowledge and the only evil is ignorance.”

- **Socrates**

(Greek philosopher)

(469 BC - 399 BC)

This chapter describes the various techniques used by the study to build the domain ontologies. Building domain ontology is a complex process and consists of various sub-tasks some of which can be automated with carefully crafted technical implementations, while others requires human intervention. Moreover some of these techniques are the result of intense research while others are rule-based techniques. In this chapter the techniques described mainly fall under two major umbrellas, (1) term-relationship based keyword extraction and (2) rule-based keyword extraction. Both of these techniques extract useful terms or keywords which are later used by humans to carefully place in an ontology model.

9.1 Term-relationship Based Extraction

Term-relationship plays a major role in several areas of research including document relevance, domain ontology construction and metadata extraction. As part

of this study in semi-automatic business ontology construction, term-relationships are used to facilitate the process of extracting useful terms from text-based data. Text-based data, e.g. news articles, provide valuable information about a company. This thesis reports on a learning model to learn term-relationships to extract related terms for a company. Bayesian Networks (**BN**) are constructed for individual companies to encode these term-relationships. These BNs are then used in the ontology construction process. To show the effectiveness of this learning model, the ontologies were used in the context of document relevance which results in excellent precision and recall values. It is shown that this model can be effectively used for learning term-relationships from text-based information sources.

9.2 Introduction

Learning term-relationships is considered one of the most useful steps in the context of knowledge discovery, construction of knowledge-bases (e.g. domain ontologies) or knowledge management issues. The main goal is to build business ontologies for major public companies listed on the NYSE and in other stock exchanges. These ontologies will be part of the business knowledge-base, which will be used for analysing textual information (such as corporate news sources, whitepapers or annual reports etc.) for individual companies.

To construct these ontologies for individual companies, up-to-date information was needed, in the form of useful terms e.g. from news articles available on business Web-sites. Useful terms can be learned from these sources, which in the later stages, can be incorporated into the ontology with human assistance. OWL-Lite was used for creating the ontology in this study. However that part is not relevant in this paper. This paper focuses mainly on the pre-processing of documents so that all related terms for a company (in this case) can be extracted from the corresponding news document set.

The block-diagram in Figure 9.1 provides some details of the ontology construction process. The main focus here though is the learning model (surrounded by dotted rectangle surrounding “Related Term-vector Generator”, “Bayes-Net Generator” and the “Bayesian Network”) involved. The model is used to retrieve important terms to be included in the ontology. As a proof of concept, it is shown

that by using this learning model, one can build ontologies which can be used in document relevance measurement to obtain high precision and recall values.

The approach of finding useful terms borrows ideas from query expansion or query term re-weighting. The objective here is to find "related term-vector" (defined later) for a particular company. This is an on-line learning process, which does not rely on any dictionary, thesaurus, or word-net to generate the "related term-vector", and can be thought of as query-expansion. News articles were used to learn the term co-occurrences for the companies and we used company names as query terms. This knowledge is used to build the "related term-vector" and the corresponding **BN**. This network provides an easy way to view the relational probabilities between two terms. This is an on-line learning approach. As new news articles are introduced the analysis is done on-the-fly to update the *ontology*.

9.3 Related Work

Learning term-term relationships has its roots in Information Retrieval (IR) research in the context of relevance feedback and is used mainly for query modification. The two main trends in this research are query term re-weighting and query expansion.

Harman [67, 68, 70, 69] examined these two trends in a probabilistic model. There he discussed the question of adding best possible terms [67] with the query. Term re-weighting has been investigated by Salton et. al. [119, 71] in addition to their experiments with variations of probabilistic and vector space model [118].

Smeaton and van Rijsbergen [124] investigated query expansion and term re-weighting using term-relationships. The results of these experiments were largely negative. Query expansion via Maximum Spanning Tree shows poor performance for unexpected query. The same happened using Nearest Neighbor approach. They cited as the reason behind this, the difficulty in estimating probability. This these introduced a simple way of estimating the probability for a set of documents. Two words are associated if they co-occur in a sentence or nearby-sentences. It is believed that, this is a more realistic and reasonable approach and can be viewed as learning the relationships from a set of documents.

User specific information has been used to expand query [24]. Personal con-

struct theory has been used in this [77] paper. The document set is analyzed to determine the probability of co-occurrence and is used instead of user-preference. A user-centric evaluation of ranking algorithms can be found in [56]. In [94] Ramesh worked with the SenTree model to find term-dependencies.

In [107] researchers used a logical approach to describe the relationship between a term and a document. They used a *probabilistic argumentation system* and claimed after Rijsbergen that IR systems are a form of uncertain inference. In order to be relevant to a term, a document must imply the term. The approach in this study can be seen from this angle, that each term must imply the occurrence of the related term with a certain probability. The idea of term-relationship is somewhat similar to term co-occurrence [101]. In that work, Peat and Willett explained the limitations of using term co-occurrence. But they used a more generic calculation of similar terms by three similarity measures: *cosine*, *dice*, and *tanimoto*. They used the whole document to find the term co-occurrence instead of a small region. This study posits that the way the similarity measures are formed and the way term co-occurrences are identified are too generic in nature to find any useful result. Instead of completely relying on the document statistics, if the usual constructs of natural language sentence formation can be exploited and a smaller region of terms used rather than the whole document, a better result can be achieved.

Use of **BNs** is comparatively more limited in IR research than in mainstream Artificial Intelligence (AI) or in Uncertainty in AI (UAI) literature. Luis M. de Campos [37, 38, 39] extensively used **BNs** for query expansion and IR research. Others include Turtle and Croft [130], Fung et.al. [62], Haines and Croft [65] (relevance feedback by using inference network), Neto [96], Pearl [100] and Silva [123]. Some of the researchers viewed the document as a sample of the collection of terms. Terms occur randomly in the document with some probability distribution and these distributions are not always known. Even if the real distribution is not known, document-term-relationship or term-term relationship can play a major role. The study uses **BN** because it is believed that the term-term-relationship can be learned as “effect of one term on another given the document set”. If the company name is used as the main (root) term, the relationships between this and any other linked term can be thought of as “a measure of the effect of the company name on the other term in the context of the discussion”. The idea behind this

work is to find this relationship, measure it and use this knowledge to construct the **BN**, and ultimately the ontology.

9.4 Our Contribution

Firstly, this work introduces a new way to learn term-relationships. Natural language documents are not just an arbitrary array of words. Words co-occurring in sentences or nearby sentences, relate to each other more closely than do those in two distant sentences. As described above, the reason behind the failure of term co-occurrences as studied in [101] can possibly be improved.

This study introduces a “Weighted-Sentence” (**WS**) based term co-occurrence model [48]. Secondly, it also introduces a **BN** based approach to encode the “term-relationship” with probability values directly influenced by term co-occurrence measure. These networks for each individual company have been used to create the corresponding ontologies.

The system architecture is shown in Figure 9.1 for a company. The news collection is pre-filtered for each company. We use this collection to generate the related-term vectors. Next step is the construction of the **BN**. Once complete, it is used to construct the ontology.

9.5 Our Approach

Our approach is described in three parts. In the first part the fundamentals of learning term-relationships are described. The second part sets out the construction of the (**BN**). In the third stage, the **BN** is used to construct the ontology, and it is then used to find the relevance ranking of new articles.

News Websites were crawled and news articles were filtered for an individual company. Let us assume that in a collection C there are in total M news articles $D_1, \dots, D_M$ pre-filtered for the company w . This constitutes the document collection C . So

$$C = \{D_1, D_2, D_3, \dots, D_M\} \quad (9.1)$$

As these documents can be thought of as a set of sentences, if document D_i has

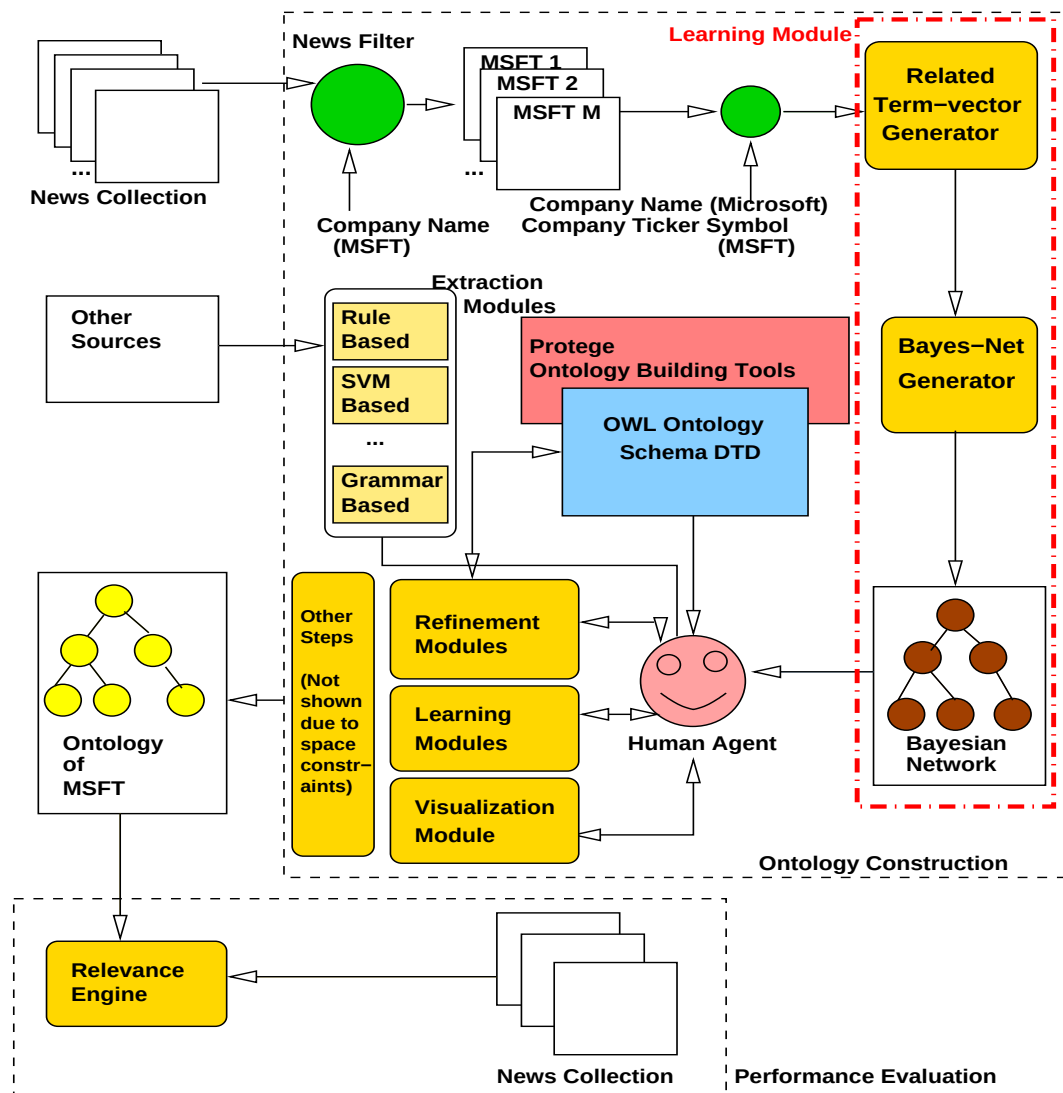


Figure 9.1. A simple architecture of our system. We create the **BN** as the knowledge base to be used in the Ontology construction. The main focus of this paper is the part, surrounded by dotted red rectangle.

N_i number of sentences in it,

$$D_i = \{S^i_1, S^i_2, S^i_3, \dots S^i_{N_i}\} \quad (9.2)$$

where $S^i_j : j^{th}$ sentence in document D_i . Similarly each sentence can be thought of as a set of words and so if S_j contains P_j number of words in it,

$$S_j = \{w^j_1, w^j_2, w^j_3, \dots w^j_{P_j}\} \quad (9.3)$$

where w^j_k represents the k^{th} word in sentence S_j .

9.5.1 Related Term-Vector

Related term-vector of a term w is a vector of all the words which co-occur with w and are important (ranked higher than a given threshold). Basically it is the set of terms for which the term-relationship measure between them and the main term is higher than a threshold. The difference between conventional co-occurring terms [101] and the approach used here is that words in the same sentence or neighbouring sentences are considered as probable candidates. At this point, it is assumed that a function Ψ will provide all the related words of w when applied to a sentence S_j containing w . Therefore,

$$\begin{aligned} & \overrightarrow{\Psi}(w, S_i) \\ &= \begin{cases} \langle \overrightarrow{w^r_1, w^r_2, w^r_3, \dots w^r_{R_i}}, & w \mathcal{R} w^r_i \text{ and } w \in N(S_i) \\ \overrightarrow{\phi}, & \text{otherwise} \end{cases} \quad (9.4) \end{aligned}$$

Ψ generates a vector of related words of the main term w . Considering the fact that function Ψ generates a vector, one can write $\Psi(w, S_i)$ as $\overrightarrow{\Psi}(w, S_i)$. $\mathcal{R}$ is a notation indicating that w and w^r_i are related. $N(S_i)$ is a function which takes the position of S_i and gives the set of neighbouring sentences (implementation details in section 9.6.1).

An example would be useful here to understand the concept. From the list of news articles pre-filtered for Microsoft, let us assume a sentence “*Microsoft (MSFT) CEO Bill Gates announced today the upcoming version of windows oper-*

ating system, codenamed longhorn". The words/phrases like "CEO", "Bill Gates", "today", "upcoming", "version", "windows", "longhorn" etc. could be all related to the keyword "Microsoft" or "MSFT". Given this sentence, the function Ψ will generate the related word/phrase vector. Now the type of this relationship (notation $\mathcal{R}$ is used to express "relationship") such as CEO_OF or PRODUCT_NAME etc. can be attributed by human agent in the next step.

Similar to the above derivations (1, 2, & 3) we can see that in case of a whole news document D_i and for the whole document collection C in category C ,

$$\vec{\Psi}(w, D_i) = \sum_i^{N_i} \vec{\Psi}(w, S_i), \quad \vec{\Psi}(w, C) = \sum_i^M \vec{\Psi}(w, D_i) \quad (9.5)$$

The algorithm to implement the Ψ function and the related-vector generating process are described in subsection 9.6.1.

9.5.2 Bayesian Network

BN is one of many graphical models available. It is simply a directed acyclic graph (DAG) where the nodes represent a set of random variables and the directed edge from one (source) node to another (destination) node represents that there is a direct influence from the source to the destination node. Our **BN** reflects the term-relationship hierarchy in a list of news articles for a company. We have a **BN** for each company and the company name (for which the pre-filtering was done) is taken as the root of the **BN**. In **BN** literature, edges represent direct influence from one node to another. Similarly here, an edge represents a similar influence regarding relationship in a sentence-set in any of the news article in the list. That, in simple notations, means that if w is at the root node, and $w\mathcal{R}w'$, then w' occurred with w or in another way, occurrence of w has a direct influence on the occurrence of w' given the news collection C .

The construction of the **BN** is described here (Figure 9.2). Here w is placed at the root node. The related term-vector is found for w and place all the related words (or R_w) under it as the children nodes. R_w is defined as

$$R_w = (w_i | \|w_i \uparrow \cdot \vec{\Psi}(w, C)\| \neq 0) \quad (9.6)$$

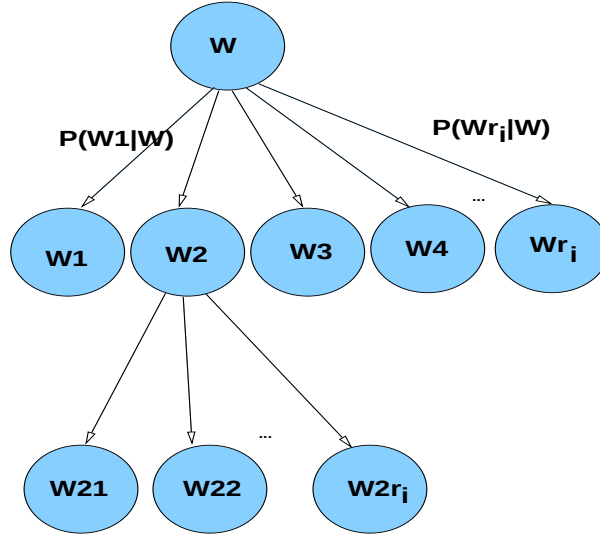


Figure 9.2. The BN structure used in our algorithm. Note that to ensure that we construct the DAG we are not connecting w_1 and w_2 even if there is a dependence between them. From w_2 's related term-vector we remove any parent or sibling node before creating the child of w_2 .

where $w_i \uparrow$ is a unit vector in the direction of $\vec{w}_i$. The conditional probability value of w_i represents the probability or relationship of w_i on w , given the news collection C or

$$Pr(w_i|w, w\mathcal{R}w_i, C) = \frac{\|w_i \uparrow \cdot \vec{\Psi}(w, C)\|}{\sum_i \|w_i \uparrow \cdot \vec{\Psi}(w, C)\|} \quad (9.7)$$

9.5.3 Ontology Construction

Once the **BN** is complete, it represents a comprehensive knowledge-base for a company. Domain experts are then used to assign labels to each individual links as “CEO”, or “PRODUCT_NAME”, or “COMPETITOR” etc. These terms and the link-types are then used to create an ontology of the company.

9.6 Implementation

This section describes the way to form the related term-vector based on term relationship. Also described is the algorithm to build the **BN** for a company. ““

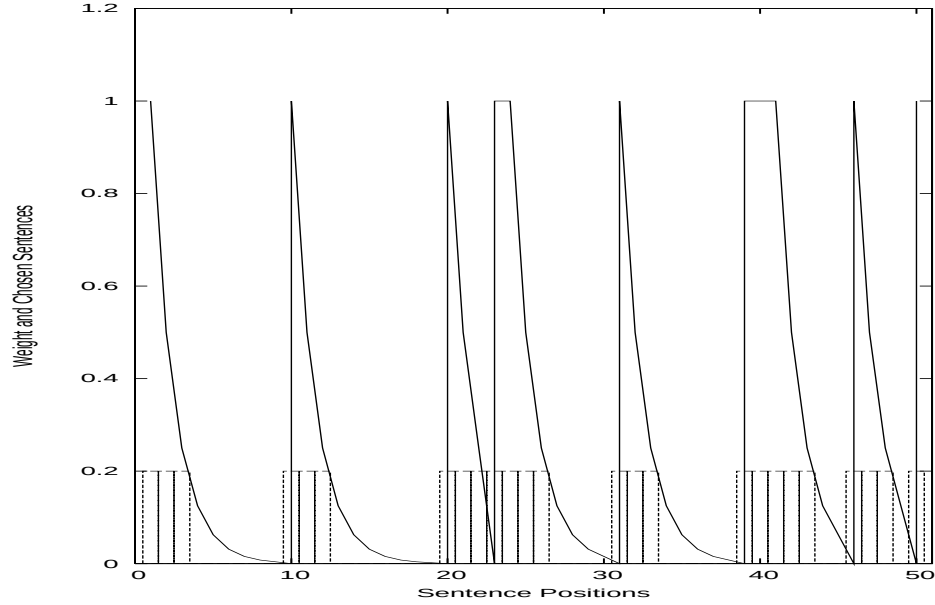


Figure 9.3. The weighting function $W(j)$ and the set of sentences included in $N(S_i)$ as shown by the bar graphs. The preliminary sentence set was 1, 10, 20, 23, 24, 31, 39, 40, 41, 46, and 50 and the final extended sentence set is 1-3, 10-12, 20-26, 31-33, 39-43, 46-48, and 50. Each bar represents a sentence.

9.6.1 Weighted Sentence Based Related-Term Vector

A novel concept of finding related-term vectors is introduced based on weighted sentences. On-line publishable natural language texts are almost always written in a coherent way; consecutive sentences are devoted to describe the topic. This simple, natural, and powerful editing style can be exploited to extract related-term vector by using the weighted sentence (WS) based method.

As mentioned earlier, related-term vector depends on the concept of term co-occurrences. Let us proceed step-by-step assembling all the concepts necessary to get the generating function for related-term vector. First, one can modify the concept of “sentences” to an “extended sentence-set”. A function $N(S_i)$ is introduced which takes the position of a sentence S_i and generates the extended sentence-set. First the following function is defined to get the weighting factor.

$$W(j) = e^{(j-i)\log(\tau)} \quad (9.8)$$

where $w \in S_i, j \geq i, \tau = \text{threshold}$. Here $W(j)$ is a weighting factor for all consecutive sentences at position j after the sentence i containing the query w . In the implementation, $\tau = 0.5$. With this definition, $N(S_i)$ is defined as

$$N(S_i) = \{S_j | j \geq i, W(j) \geq \epsilon\} \quad (9.9)$$

In our implementation $\epsilon = 0.2$. For a typical document, sentences at position 1, 10, 20, 23, 24, 31, 39, 40, 41, 46, and 50 contain the query term, and therefore the $W(j)$ will look like Figure 9.3. In this particular case sentences at positions 1-3, 10-12, 20-26, 31-33, 39-43, 46-48, and 50 will be included in corresponding $N(S_i)$ s. Term-relationships are calculated using a formula similar to [106], where term-term relationships were calculated per-document basis. This study calculates it per-collection basis. The significance of a word w_m co-occurring with w in sentence S_i is

$$\sigma_{w_m} = \frac{ptf_m}{\sqrt{\sum_r ptf_r^2}} \times \log \Phi \quad (9.10)$$

where

$$ptf_m = \left(\frac{tf_m}{\max_r tf_r} \right) \quad (9.11)$$

and where

$$\Phi = \frac{N}{n_m} \quad (9.12)$$

where

$$tf_m = ntf_m^j \times W(j | S_j \in N(S_i)) \text{ (from (9))} \quad (9.13)$$

where $ntf_i^j =$ term-frequency of term w_m in sentence S_j . N is the number of total sentences in the document collection C which is (from (1) and (2))

$$N = \sum_k^M N_k, \quad n_m = \sum_{S_k \notin N(S_i), w_m \in Nouns(S_k)}^M N_k \quad (9.14)$$

which in simple term is the number of sentences outside the set $N(S_i)$ where the term w_m appears. $Nouns(S_k)$ is an NLP function which gives the set of all nouns of a sentence, taken as its input. Now $\mathcal{R}$ represents the relatedness between two terms. It is defined as a relation between w and w_m which produces the candidacy

of w_m to be included in the related-term vector depending on some conditions.

$$\mathcal{R} \equiv \{f : w \rightarrow w_m | \sigma_{w_m} > \zeta, w_m \in Nouns(N(S_i))\} \quad (9.15)$$

Here $Nouns(N(S_i))$ gives the whole set of nouns from the extended WS set of S_i . ζ is a threshold.

9.6.2 BN Constructor Algorithm

Once the related term-vector is formed, the **BN** for a company is built using equation (7) and (8). After the first level is constructed, a recursive process was used to build the rest of the tree. It is described in algorithms 9 and 10. Here the corresponding node for w is denoted as $t(w)$ and the tree is denoted as $\mathcal{T}$

```

Input   : News collection  $C$ , Company name  $w$ 
Output:  $\mathcal{T}$ , a BN for word relations rooted at  $w$ 
begin
    //Initialization - Create the root node  $t(w)$ 
     $t(w) \leftarrow \text{new TreeType}()$  //Root node Create
    //Assign probability of 1 to  $t(w)$ 
     $Pr(w) = 1$  //Assign probability of 1 to  $t(w)$ 
    //  $\mathcal{T}$  is a TreeType Node
     $\mathcal{T} \leftarrow t(w)$  //TreeType node
    //  $\mathcal{N}$  is a set of structures (word,level)
     $\mathcal{N} \leftarrow (w, 0)$  //(word,level)
     $l \leftarrow 0$ 

    //Traversal and new node addition
    while ( $l < l_t$  or No new node can be added) do
        | Traverse  $\mathcal{T}$  in BFS fashion
        | AddNewNode( $\mathcal{T}, l$ )
    end
end

```

Algorithm 9: The **BN** building process. $\mathcal{N}$ is a structure which we use for storing the word and the level of the **BN** tree in which the word is coming inside a node. TempSet keeps the R_w which is defined in equation (7). The formation of TempSet ensures that the network is a DAG. The result of using this algorithm for “eBay” has been shown in figure 9.4.

```

Input   : News collection  $C$ , Company name  $w$ 
Output:  $\mathcal{T}$ , a BN for word relations rooted at  $w$ 
begin
  Function AddNewNode
    for (each node  $t(w)$  in level  $l$ ) do
       $TempSet \leftarrow R_w$  //  $R_w$  is defined in equation(7)
      for (each word  $w_k \in R_w$ ) do
        if  $((w_k, l_k) \in \mathcal{N} \text{ and } l_k \leq l)$  then
          Delete  $w_k$  from  $TempSet$ 
        end
      end
      for (each word  $w_i \in TempSet$ ) do
        //Create the child node  $t(w_i)$ 
         $t(w) \leftarrow \text{new TreeType}()$ 
        //Assign probability From equation (7))
        //From equation (7))
         $Pr(t(w)) = Pr(w_i|w, w\mathcal{R}w_i, C)$ 
         $\mathcal{N} \leftarrow \mathcal{N} \cup (w_i, l + 1)$ 
        Add  $t(w)$  to  $\mathcal{T}$ 
      end
    end
  end

```

Algorithm 10: (contd..) The BN building process. $\mathcal{N}$ is a structure which we use for storing the word and the level of the BN tree in which the word is coming inside a node. TempSet keeps the R_w which is defined in equation (7). The formation of TempSet ensures that the network is a DAG. The result of using this algorithm for “eBay” has been shown in figure 9.4.

Lemma 1 The formation of TempSet ensures the construction of a BN

This is true because $\mathcal{N}$ is a set which keeps track of all the existing nodes by including in the set the word associated with the node and the node level of the corresponding node in the tree. Now, each time the algorithm traverses the tree in BFS fashion and tries to add the nodes for the related word of any word, it takes the related word from the word set (TempSet) for which the label is not defined OR if it is there, then at least it is not in the upper level of the tree. Because of

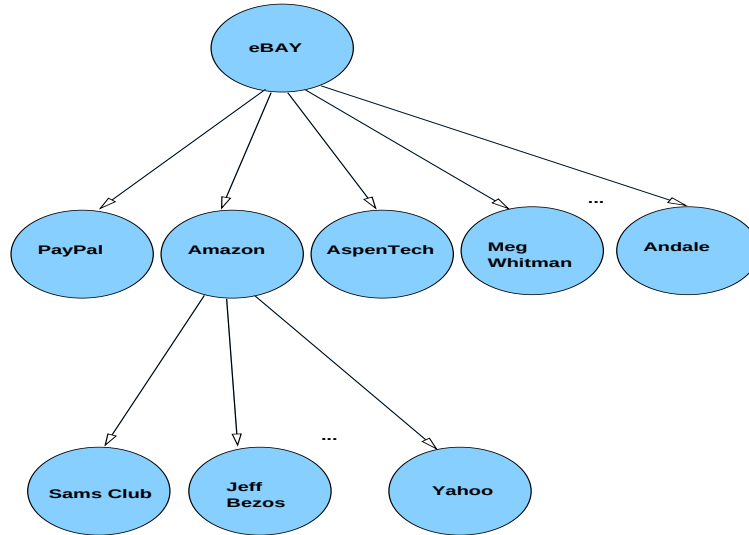


Figure 9.4. The **BN** created by our algorithm for eBay. Note that “Paypal”, “Amazon”, “AspenTech”, “Meg Whitman”, etc. are all related with “eBay”. We show the sub-nodes only for “Amazon” and as we can verify “Sams’ Club”, “Jeff Bezos”, “Yahoo” etc. are also co-related with “Amazon”.

this no edge forms between a word node and any previously added word nodes in the higher level and the directed acyclic nature of the graph is maintained.

In simple terms, the **BN** is constructed starting from the root node as company name. Then we take each of the terms in the related term-vector of the company name. The conditional probability values are calculated according to equation (7). Once this step is complete we use the same technique for each of these terms and find the related term-vectors for them.

9.6.3 Experiment and Result

We implemented above algorithms in Perl on Unix platform. We used the Alembic workbench [91] for the *Noun* function. Alembic has a nice feature of producing noun phrases rather than a single noun word. This feature is useful in case of person names or company names consisting of multiple words.

There is no definitive way of showing the superiority of one ontology over another. This is more true due to the fact that ontologies are constructed for different purposes, so there is lack of benchmark ontologies and benchmark data and application set, which can be used. However, in the next chapter more concrete proof of the effectiveness of these ontologies will be given.

However, in this chapter one example **BN** for eBay Inc. (EBAY) in Figure 9.4 is shown. From this figure it is vividly clear that our construction technique works pretty well and consistent with knowledge about eBay.

Now, to show the effectiveness of this approach, **BNs** for a few companies are used to find document relevance for the corresponding news article set. The results are reported in Table 10.1 in the next chapter. As we see from this table, this method outperforms the basic vanilla TF-IDF measures. Actually these results depend on how many useful documents are thresholded, and so TF-IDF gives perfect recall as it includes all the news article (based on company ticker symbols or name which is present in all these documents – as they are already pre-filtered from Yahoo and other Web-sites) and so provides no guarantee of precision performance. Otherwise, according to the use of ontologies, it only picks up the useful and more relevant ones and so precision is much better than the TF-IDF case and that way it outperforms the TF-IDF in overall F-measure.

9.6.4 Using the **BN** to construct Ontology

As described earlier, the **BN** in this study was created using the values in the conditional probability table. In this step domain experts are used to label the edges. Once the labelling is done, a converter algorithm can be used to fill up the OWL-Lite instances. In the purpose of this thesis and to use these ontologies easily, we actually filled up XML [6] instances from these **BNs**. As demonstrated in the next few sections, finding keywords only from the news articles is not enough for the purpose of this whole work. Various other information is needed which can be available in the company profile pages, company competitor pages and other informative documents.

9.7 Rule-based Extraction

As described in the block diagram in Figure 9.1, data have been collected from various sources. This also includes company profile pages and other related information sources crawled, collected, cleaned and finally parsed to extract the right information. Figure 9.5 shows one example of the company GE (General Electric) from Yahoo profile page.

Some intelligent processing of this information was done. However due to the space constraints of this thesis, and to the focus on the main topic of thesis we will not evaluate any of these ideas. However, we are confident to claim that applying each of these techniques improved the overall performance.

9.7.1 Company Ticker Symbol

The company ticker symbol was used as a query keyword to determine the basic relevance of the news article as well as individual sentences. This will be elaborated more in the “sentence-based explanation” model in Chapter 11.

9.7.2 Company Name and Variations

Alongside the main ticker symbol, the company name was also used to increase the relevance measure. Company names, when abbreviated, such as “Intl. Bus. Machines”, “International Business Machines”, or IBM, should all be considered as they refer to the same company. A database of the company names was created semi-automatically (with the help of some grammar rules and hand-coding and entry of the names in the database). It improved the overall efficiency.

9.7.3 Company Product Information

Company product names are another important piece of information to be used in the relevance model. An example would be appropriate here. A news article with no mention of “MSFT” or “Microsoft”, but lots of reference to “Windows XP” or “Longhorn” is as equally important to Microsoft as any other news article having the company name. So product names and their abbreviations play a major role in

GE: Profile for GEN ELECTRIC CO - Yahoo! Finance - Mozilla

File Edit View Go Bookmarks Tools Window Help

MSN Hotmail - Mes... file:///exp...hort-model Microsoft Worldwid... Candidate Portal - ... GE: Profile for GEN ...

General Electric Co (GE) At 11:21AM ET: **33.78** ↓ 0.23 (0.68%)

MORE ON GE

Quotes
[Summary](#)
[Real-Time ECN](#)
[Options](#)
[Historical Prices](#)

Charts
[Basic Chart](#)
[Technical Analysis](#)

News & Info
[Headlines](#)
[Company Events](#)
[Message Board](#)

Company
 Profile
[Key Statistics](#)
[SEC Filings](#)
[Competitors](#)
[Industry](#)
[Components](#)

Analyst Coverage
[Analyst Opinion](#)
[Analyst Estimates](#)
[Research Reports](#)
[Star Analysts](#)

Ownership
[Major Holders](#)
[Insider](#)
[Transactions](#)
[Insider Roster](#)

Financials
[Income Statement](#)
[Balance Sheet](#)
[Cash Flow](#)

Fidelity Brokerage Services
 New **low** commissions
www.fidelity.com/tradesmarter

Profile Get Profile for: GO

General Electric Co
 3135 Easton Turnpike
 Fairfield, CT 06828-0001
 Phone: 203-373-2211
 Fax: 203-373-3131
 Web Site: <http://www.ge.com>

DETAILS

Index Membership:	Dow Jones Composite Dow Industrials S&P 100 S&P 500 S&P 1500 Super Comp
Sector:	Conglomerates
Industry:	Conglomerates
Employees (last reported count):	NaN

BUSINESS SUMMARY
 General Electric Company engages in the development, manufacture, and marketing of various products for the generation, transmission, distribution, control, and utilization of electricity. The company operates through 11 segments: Advanced Materials, Commercial Finance, Consumer Finance, Consumer and Industrial, Energy, Equipment and Other Services, Healthcare, Infrastructure, Insurance, NBC Universal, and Transportation. Through these segments, it delivers network television services; operates television stations; and provides cable, Internet, and multimedia programming services. In addition, the company offers various financial and other services, including consumer financing, commercial and industrial financing, real estate financing, asset management and leasing, mortgage services, consumer savings and insurance services, and specialty insurance and reinsurance. GE's products include major appliances; lighting products; industrial automation products; medical diagnostic imaging equipment; motors; electrical distribution and control equipment; locomotives; power generation and delivery

ADVERTISEMENT

Fidelity
 Smart move™

Need a reason to buy index funds?
 How about 10.

Index funds from Fidelity
 for ten basis points.

[Click here to learn more.](#)

Please carefully consider the fund's investment objective, risks, charges and expenses applicable to a continued investment in the fund before investing. For this and other information, call or write to Fidelity for a free prospectus, or view one online. Read it carefully before you invest or send money.
 Fidelity Distributors Corporation 358600
 1.800.FIDELITY

CORPORATE GOVERNANCE NEW!
 General Electric Co's [Corporate Governance Quotient \(CGQ®\)](#) as of 1-Aug-05 is better than **83%** of S&P 500 companies and **95.6%** of Capital Goods companies.
 Brought to you by [Institutional Shareholder Services](#).

KEY EXECUTIVES

	Pay	Exercised
Mr. Jeffrey R. Immelt , 49 Chairman, Chief Exec. Officer and Member of Public Responsibilities Committee	\$ 8.30M	\$ 3.63M
Mr. William M. Castell , 56 Vice Chairman, Exec. Officer, Member of Public Responsibilities Committee, Chief Exec. Officer of GE Healthcare and Pres of GE Healthcare	\$ 4.26M	N/A
Mr. Dennis D. Dammerman , 59 Vice Chairman, Member of Public Responsibilities	\$ 7.95M	\$ 1.05M

Figure 9.5. A profile page for the company GE (General Electric). We collected useful information from pages like these to build the ontologies of the company.

identifying the right article with the right amount of relevance assigned according to them.

9.7.4 Company Executives

Company executives' names are a useful source of information which can be efficiently used to get relevant documents/sentences. For example, any sentence with the phrase "Bill Gates" is sure to be pertinent to Microsoft corporation. This is more prominent in financial news articles as the company executives' names come more often in these articles.

9.7.5 Synoname

Synonames are usually applied in the context of archiving historical information. They are used to find different versions of personal names which vary greatly from one institution to the next, reflecting traditional differences¹. There is a computer program named **SYNONAME**, which automatically matches many possible forms of a single personal name by using an ordered sequence of 12 algorithms for pattern matching including both character and word-matching. The matched pairs of these names are considered to be "candidate matches" until confirmed by a human name-authority editor. Running against a merged file of artists' names from museum collections data, the program performed with an accuracy rate of 97.4% and an optimum efficiency rate of 90.8%. Accuracy can increase to nearly 99% at the expense of some efficiency.

In the absence of an available version of this program's download-able version, the name matching engine was implemented by simple regular expression matching and as it is using pre-edited well-tabulated data, it worked with over almost 100% accuracy. Table 9.1 shows part of the table which was edited after collecting the information for mostly Christian names used in the USA^{2, 3, 4}

¹<http://crl.nmsu.edu/cgi-bin/Tools/CLR/clrinfo?SYNONAME>

²<http://freereg.rootsweb.com/howto/realnames.htm>

³<http://freereg.rootsweb.com/howto/latinnames.htm>

⁴<http://www.hampton.lib.nh.us/hampton/history/dow/abbrev.htm>

Name	Synoname	Name	Synoname
Abraham	Abra, Abram	Anthony	Ant, Tony
Benjamin	Ben, Benja	Charles	Charlie, Cha
Christopher	Chris, Christo	Edmund	Ed., Edm
Francis	Franc, Ffran	Jeffrey	Geoffrey, Jef
...	...	...	...
John	Jo	Nathaniel	Nate, Nathl, Nathan
Robert	Rob, Bob	Samuel	Sam, Samu,
Thomas	Tom, Thom	William	Bill, Billie

Table 9.1. Part of the Synoname table used for extracting names abbreviated or used other ways.

9.7.6 Competitors

Competitors' names are also very useful in the context of document relevance. Competitors names can indicate that the article is about the group of companies in the same sector, and thus can be inferred as useful to the company. For example, an article under the MSFT (Microsoft) bin may have a sentence about AAPL (Apple) or IBM (IBM), and as all these companies are in the Software business, the sentence will be deemed relevant in the context of MSFT. Though this feature was not extensively used in the recent version of the software in this study, it can be used more fruitfully in future. Chapter 12 includes further discussions and future directions, where we conclude the thesis..

All these techniques are useful in extracting more relevant information from articles. In the next chapter, as seen in Table 10.1 and Figures 10.1, 10.2, and 10.3, all these techniques can improve the document and sentence level relevance and the overall keyword-model.

9.8 Conclusion and Future Work

This study developed a novel learning method to measure term-relationship from a collection of news articles. Once the term-relationships are learned, and the related term-vector has been found, this knowledge is provided to domain experts in the form of a Bayesian Network, who then label the relation-types using the ontology schema. The effectiveness of our learning method is demonstrated in the context

of document relevance measure in the next chapter.

Chapter 10

Improving the Keyword Model

“Be not ashamed of mistakes and thus make them crimes.”

- **Confucius**

(Chinese philosopher, politician, and reformer)

(551 BC - 479 BC)

Domain knowledge is now ready to be used in the form of domain ontologies. These ontologies will be used to find relevant documents, important sentences inside these relevant documents and henceforth be used to improve upon the results received by using the basic keyword-model on a handful of companies (in Chapter 8).

As noticed in the previous chapter, building a domain ontology is a complex procedure. We built individual ontologies for individual companies, but this was not accomplished in just one step. Old news articles, and relevant documents were analyzed to extract useful keywords. Then the relationship was visualized by using **Bayesian Network**. In the following step, these keywords were put in their proper places (according to the type of relations). Rule-based techniques were also used to extract other useful information about these companies wherever available. This includes the company profile page, company competitor page, and use of other

ad-hoc mechanisms such as extracting and using word synonym and synonames for company officials.

10.1 Preliminary Improvement

Several improvements resulted from using all these techniques. The first few tables and associated figures show how the use of ontologies can improve the document-level and sentence-level relevance.

10.2 Document-level Relevance

We try to give a measure of document relevance with the use of ontologies and how it can be useful in the context of document relevance. To find the relevance measure of a document D_i for the company w all the terms of the **BN** rooted at w are used. From equation (2) in the previous chapter,

$$D_i = \{S^i_1, S^i_2, S^i_3, \dots S^i_{N_i}\} \quad (10.1)$$

Using the conditional probability values of each term in the **BN** with the root probability value as 1, each sentence is marked with a significance value of $\mathfrak{S}(S_j)$. That means if there is a word w_k in the **BN** with p parent nodes then the significance of the term w_k is calculated as

$$\begin{aligned} \mathfrak{S}(w_k) &= \sum_{p_i}^p P(w_k|w_k^{p_i}) \times P(w_k^{p_i}) \\ &= \sum_{p_i}^p P(w_k|w_k^{p_1}) \times P(w_k^{p_1}|w_k^{p_1p_1}) \times \dots \times 1 \end{aligned} \quad (10.2)$$

Sentence significance is calculated as

$$\mathfrak{S}(S_j) = \sum_{w_k \in S_j} \mathfrak{S}(w_k) \quad (10.3)$$

which means that for all the terms w_k in sentence S_j if it is present in the **BN**, the equation (15) is used to calculate the individual significance values for these

terms and they are simply added to measure the sentence significance. $\mathfrak{S}(S_j)$ can be greater than 1. $\mathfrak{S}(S_j) = 0$ if there is no term in S_j which is in the network. Let us assume that there are total Y_i sentences (Y=Yes) in D_i which has significance value greater than 0 (we can identify these sentences as S_{yj}^i). Then the document relevance is defined as

$$\mathfrak{S}(D_i) = \mathfrak{P}\mathfrak{S}(S_{yj}^i) \times \frac{Y_i}{N_i} \times \frac{Y_i}{\sum_{D_i}^C S_{yj}^i} \quad (10.4)$$

where

$$\mathfrak{P}\mathfrak{S}(S_{yj}^i) = \frac{\sum_{\forall S_{yj}^i} \mathfrak{S}(S_{yj}^i)/Y_i}{\sqrt{\sum_{D_i}^C (\sum_{\forall S_{yj}^i} \mathfrak{S}(S_{yj}^i)/Y_i)^2}} \quad (10.5)$$

The above equation (15) ensures that the number of sentences having significance value > 0 and the percentage of such sentences an article contributes to the whole list are listed into account.

10.2.1 Data Set for Document-level Relevance

From our main dataset, 112 stock symbols were selected for this experiment. Lists for 14 companies are taken to set up a user-survey. However the survey did not produce any good amount of user turnout and so we could not definitely claim the superiority of using ontologies. However for a few cases, the document level relevance turned out to be better than vanilla TF-IDF.

10.2.2 Evaluation

To evaluate the relevance of the news collection for the company, the precision and recall values are used as the metric. To get user feedback for their relevance we developed our own web-based interface and conducted a user-survey for 4 months. Actually the response was poor, but whatever minimal response was collected, was good in quality. The rankings given by them for these news articles were collected.

Symbol	Our Peci- sion	Our Re- call	Our F- measure	TF-IDF Peci- sion	TF-IDF Recall	TF- IDF F- measure
AAPL	0.90	0.94	0.919	0.581	1.0	0.734
EBAY	0.927	0.955	0.94	0.62	1.0	0.765
MSFT	0.982	0.99	0.9859	0.92	1.0	0.958

Table 10.1. Details of the result. Precision from using ontologies are shown in the second column followed by Recall and van-Rijsbergen F-Measure. Precision, Recall and F-Measure using plain vanilla TF-IDF are shown in fifth, sixth, and seventh columns.

10.2.3 Metric

In traditional relevance measure, precision is defined as the ratio of the number of relevant items r found and the total number of items t found. $Precision = \frac{r}{t}$ Recall has been defined as the ratio of the number of relevant items found and the desired number of relevant items. The desired number of relevant items includes the number of relevant items found and the missed relevant items m . $Recall = \frac{r}{r+m}$ van-Rijsbergen F-Measure which is the harmonic mean of precision and recall, which is

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (10.6)$$

In our experiment r is the number of documents for which user-given relevance is more than the threshold (set at 0.3) which is used to ensure that the effect of pre-filtering is eliminated.

10.3 Sentence-level Relevance

Using ontologies in different steps can be used to extract more relevant sentences in an article. In this section it is shown that the gradual use of more and more knowledge can bring more and more relevant sentences and increase the recall values and thus the overall F-measure of the sentence-level extraction. It was decided to use Microsoft's (MSFT's) news article-set in this context. We took 1000 Microsoft news articles and applied our sentence extraction algorithm to extract possible sentences which are relevant to Microsoft.

In the first stage only the ticker symbol of Microsoft or MSFT was applied. In

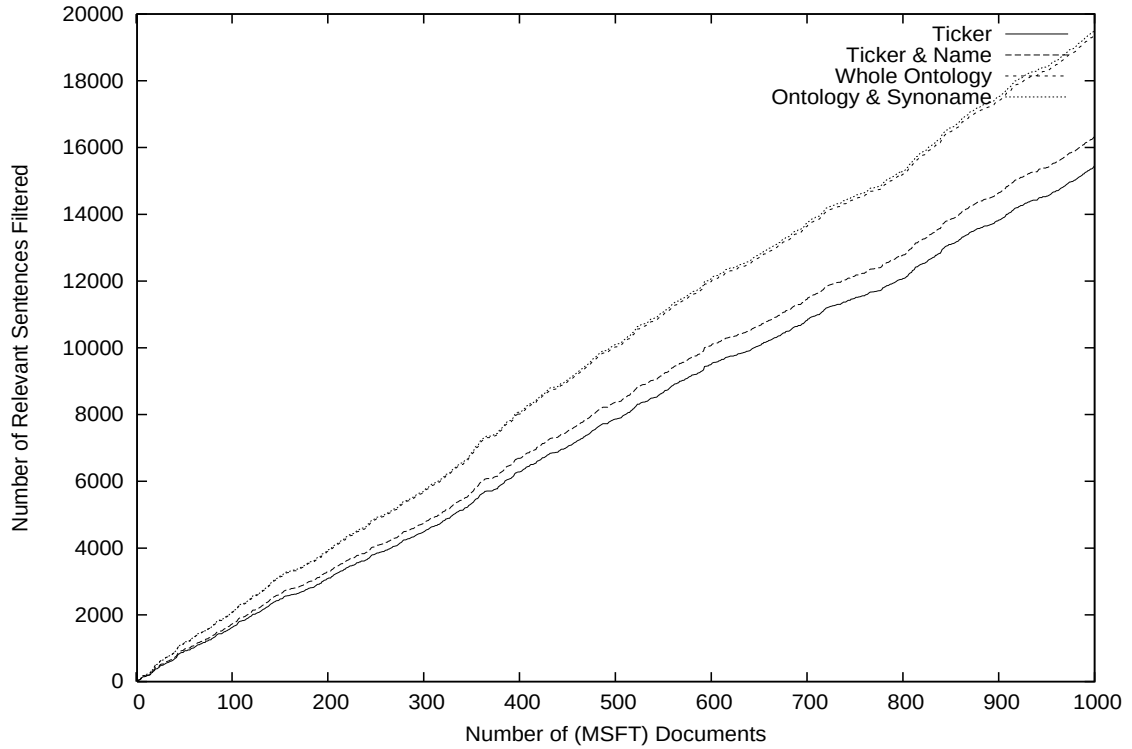


Figure 10.1. This figure shows the number of relevant sentences extracted from around 1000 news articles for the company MSFT or Microsoft. We gradually increase the knowledge base capacity from just the ticker symbol to ontology with the use of synonyms. As we see the number of relevant sentences extracted increase dramatically by using more and more relevant information.

the second stage the ticker symbol as well as the name “Microsoft” were applied. In the third phase the whole Microsoft ontology, as we built it, was applied and in the fourth stage the extended version of the ontology with the use of synonyms was applied.

As seen from the Figures 10.1, 10.2 and 10.3, use of ontology with the use of synonyms performed the best among all the other choices. Therefore, it is proved that when this combination is used, high recall value of relevant sentence extraction will result.

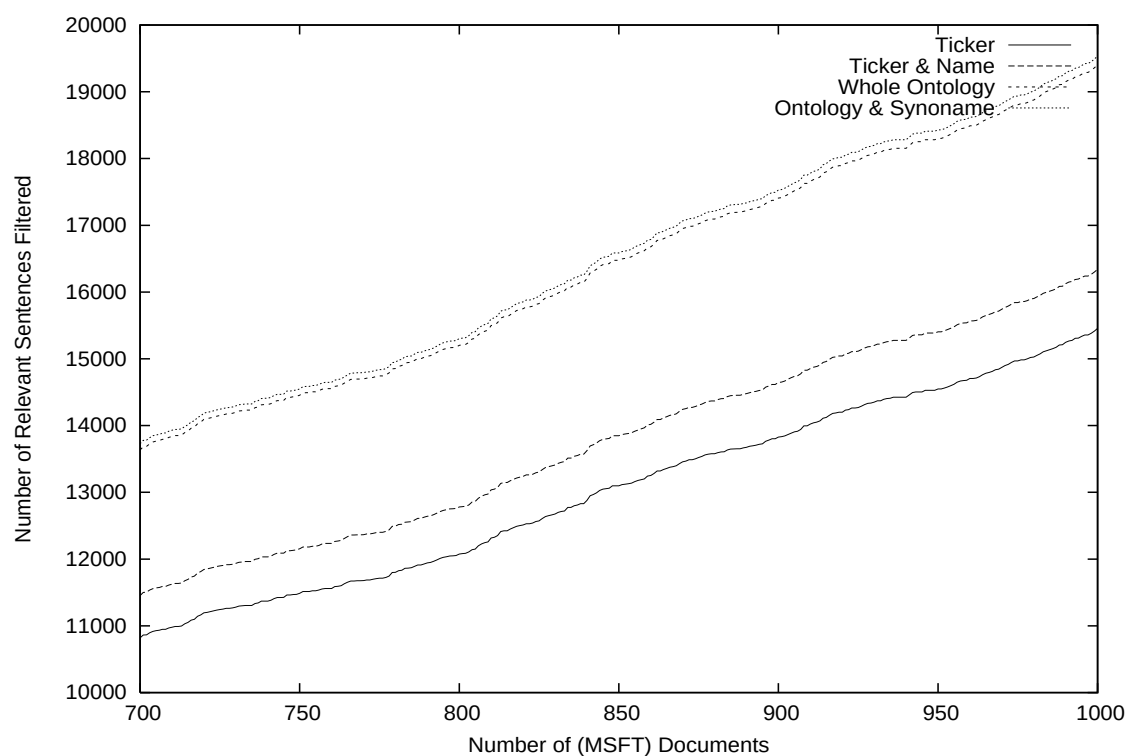


Figure 10.2. This figure shows part of the Figure 10.1 blown up to show the effectiveness of using synonyms for the company executives.

10.4 Improving Keyword Model Performance

This section shows how keyword model performance can be improved by two methods: (1) using ontologies and (2) using a variable window approach.

10.4.1 Using Ontology

Ontologies are used to improve the performance in the keyword model for all the 20 companies as shown in the Table 8.1, and the results look more promising [49]. The performance improvement is shown in Figure 10.4. From this figure it is seen that in some cases the use of ontologies did not result in very large increase in performance. When we looked for the reason behind this, we found that the use of ontologies in some cases restricts the amount of information available for analyzing and calculating the entropy. However, other than this, the performance

Ticker	Date	Price Swing	Real Cause/Fact/Analysis	Keywords Generated
EBAY (eBay)	Mar 14, 2005	DOWN	Expected federal court ruling of a suit against eBay by MerceExchange Inc. (started in September 2001) and in the last few cases MerceExchange was awarded \$5 million and \$29 million.	merceexchange, appeals, consignment, federal, claim, multiple, award, court, auction
EBAY (eBay)	Feb 13, 2005	UP	There was a talk in the town that the PowerSellers – members of a voluntary program ebay offers to its high-volume customers – will leave eBay completely or substantially reduce the listing. However “there appears to be no PowerSeller interest in leaving ebay, but there is interest in allocating incremental investment dollars to other channels, specifically sellers’ own websites,” - Mark Mahaney (Analyst)	powerseller, bad, shopper, voluntary, multiple, dollars, amazon, executive, offer
GE (General Electric)	Jul 15, 2004	UP	Sales level implies growth of 8 “When you can maintain that level of ROE for 10 years or more it shows that a company has a sustainable competitive advantage,” says Jensen portfolio manager Robert Millen. John Joyce (since named to head up the company’s services arm) said customer infrastructure was the oldest it’s been in nearly two decades, hinting buying would soon accelerate	level, economic, accelerate, joyce, growth, portfolio, john

Table 10.2. Some instances from the keyword model for various companies. As we see keywords are extracted which are very relevant to the facts pertinent to the possible reasons for the price swings for these companies. These results are mainly from variable window approach with the use of ontology (**VW + Onto**)

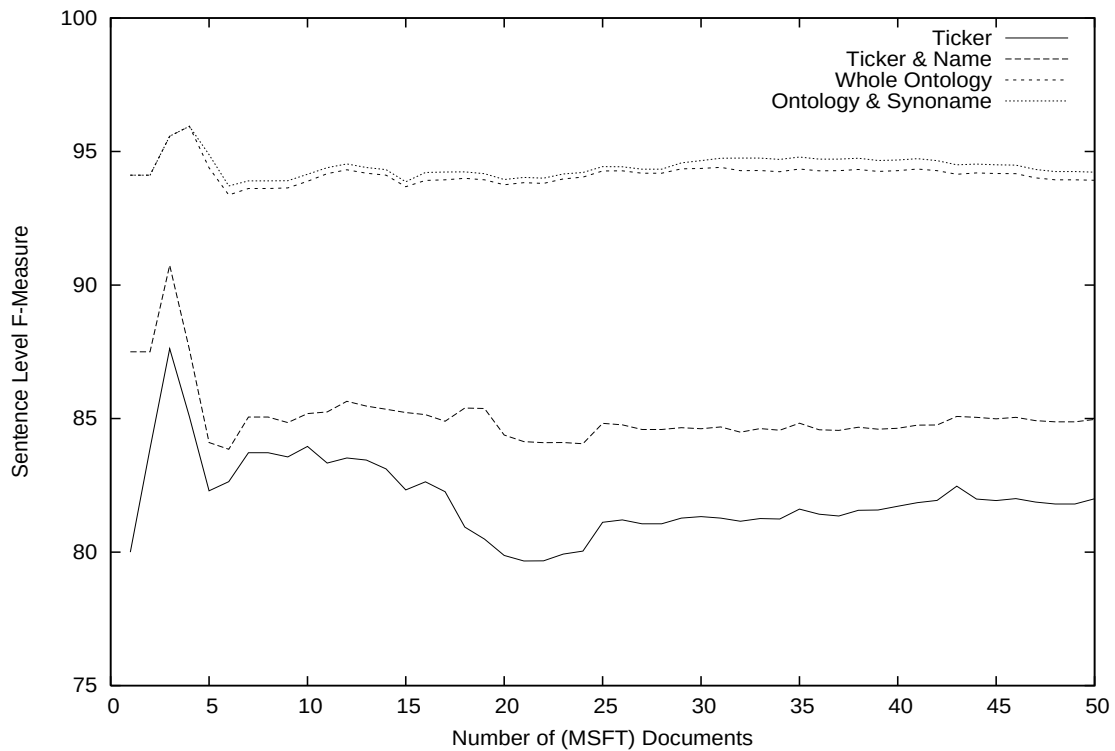


Figure 10.3. This figure shows the F-measure for the first 50 news articles from the above chart. The precision and recall values were measured for all the stages. This shows that using ontology and synonym results in a much higher F-measure than use of just the ticker symbol (which the finance pages of Yahoo usually does).

can be improved by using the business ontology.

10.4.2 Using Variable Window

Variable window is a much better concept than fixed window. This was described in detail in subsection 6.1.2 in Chapter 6. Figure 10.4 shows the performance improvement. As seen from this figure, using variable window has a much larger effect on performance than using the domain ontology. Some instances of results (in the form of keyword) are also shown for different companies for a few pivotal dates in Tables 10.2, 10.3, and 10.4.

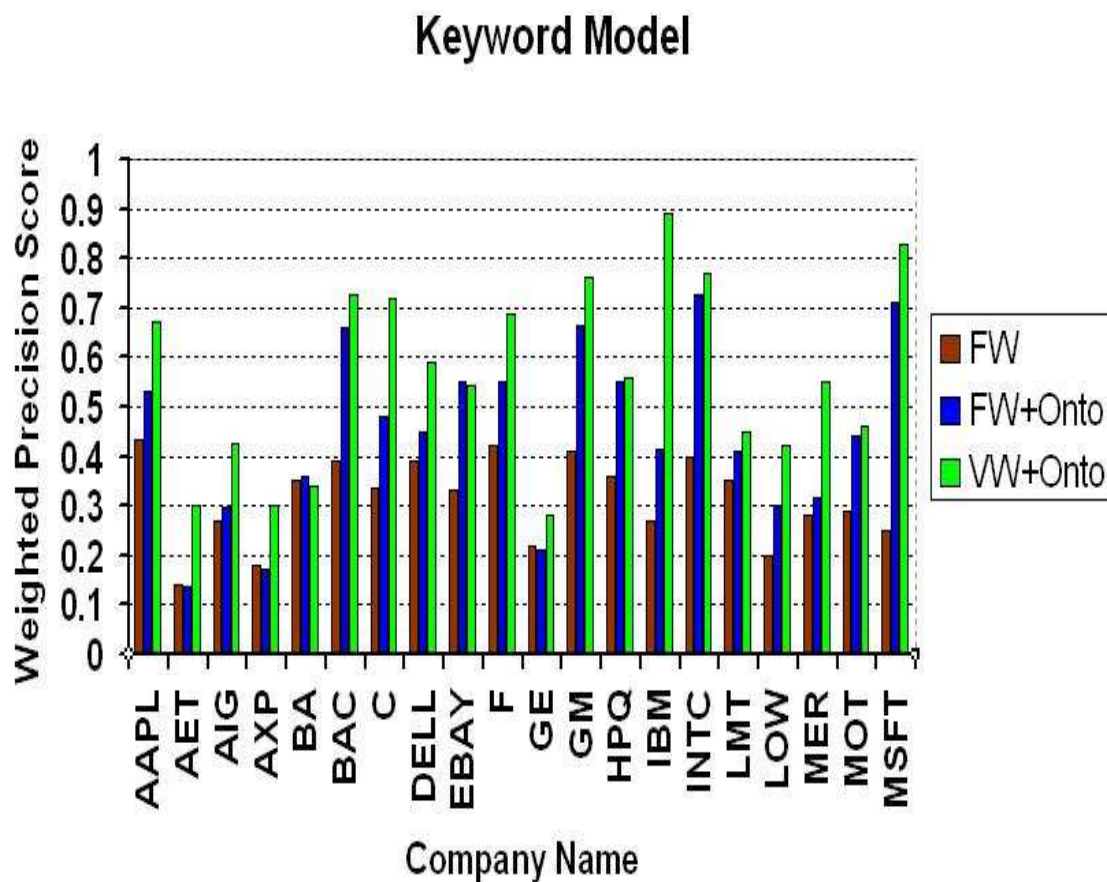


Figure 10.4. The improvement in the weighted precision scoring by using the ontology and the variable window approach. The vertical axis represents the “Score” or the “Weighted Precision Score”. The horizontal axis represents the company. **FW** represents the case for fixed window, **FW + Onto** shows the case when we apply fixed window approach with ontologies, and **VW + Onto** is the case where we used variable window approach with ontologies. **VW + Onto** outperforms the other two cases in almost all companies.

Ticker	Date	Price Swing	Real Cause/Fact/Analysis	Keywords Generated
HPQ (Hewlett Packard)	Mar 30, 2005	UP	HP issued a press release entitled “HP Names Mark Hurd of NCR to Serve as CEO and President,” a copy of which is filed with this report as Exhibit 99. “The H-P board is taking a very hands-on approach to the company now and they basically wanted someone who is skilled and talented, but also someone who will be answering to the board in a direct way,” said Charles King, principal with Pund-IT Research	mark, hurd, change, board, charles, principal, serve, fiorina
INTC (Intel)	Aug 12, 2004	DOWN	In late July Intel said it would miss a deadline for shipping its 4-gigahertz version of the Pentium 4 by one quarter	gigahertz, differentiate, lower, fall, pentium, deadline
INTC (Intel)	May 27, 2005	UP	At a gathering with analysts and reporters last week, Intel executive Gerald Holzhammer said the company is shipping 100,000 Pentium D processors this quarter - “We’re shipping 100,000 this quarter, and we’re going to ship millions by the end of the year”	indicator, gerald, ship, holzhammer, millions, pentium

Table 10.3. Some instances from the keyword model for various companies. As we see keywords are extracted which are very relevant to the facts pertinent to the possible reasons for the price swings for these companies. These results are mainly from variable window approach with the use of ontology (**VW + Onto**)

10.5 Conclusion

The keyword model is the basic model used for the work in this study and as shown in the above analysis, its performance can be improved by the use of ontologies as well as the variable window approach. In the next chapter some interesting observations in this market model are presented.

Ticker	Date	Price Swing	Real Cause/Fact/Analysis	Keywords Generated
LMT (Lock Heed Martin)	May 4, 2005	UP	lockheed martin announces first quarter 2005 results. the first quarter results include an after-tax gain of \$31 million. LMT got an order to deliver a total of 80 aircraft.	results, order, models, commander, leaders, f-16, excited
MSFT (Microsoft)	Aug 3, 2005	UP	There is a speculation that the software company may raise its bid to secure a controlling stake in Indian Banking Software firm I-Flex. “The possibility of Microsoft raising its dividend or Microsoft announcing a special dividend, and Microsoft call option volume is aggressive on dividend possibilities”,said Paul Foster, options strategist.	i-flex, raising, redact, banking, dividend
MSFT (Microsoft)	Apr 7, 2005	UP	An independent survey of 555 businesses found that Microsoft’s much maligned Windows operating system provides service and reliability that is at least as good as – and often better than – that of upstart rival Linux, and is nearly as secure. Although Linux fans are sure to deride the study as Microsoft propaganda, it was conducted by Yankee group, a respected Boston-based it research and consulting company, which paid for it internally, with no outside funding. But it certainly gives Microsoft sales people more ammunition.	propaganda, people, found, linux, ammunition, service

Table 10.4. Some instances from the keyword model for various companies. As we see keywords are extracted which are very relevant to the facts pertinent to the possible reasons for the price swings for these companies. These results are mainly from variable window approach with the use of ontology (**VW + Onto**)

Chapter 11

Interesting Observations

“There is nothing like looking, if you want to find something. You certainly usually find something, if you look, but it is not always quite the something you were after.”

- J.R.R. Tolkien

(English Writer and Author of The Lord of the Rings)
(1892-1973)

This short chapter shows some observations from the keyword-model which are very interesting and worth sharing. As shown in the last chapter, this thesis investigated 20 companies and 10 best dates for each company were taken into consideration for this study. These dates are taken from the duration Jan 1, 2004 to Aug 3, 2005 when the price swings (UP or DOWN) for individual companies were the higher than a threshold selected. If for any company, the number of dates when the price swings were above the threshold was more than 10 we took only the first 10. Therefore, we simple took the top 10 dates when the price swings were the highest. While examining these dates for keyword-based explanation, other interesting facts were discovered. This chapter describes some of them in brief.

11.1 Document Distribution Correlation

Before reporting on document distribution correlation, a definition of document distribution is necessary. It will be defined in the next subsection, and the graph of document distribution will be shown by arranging the company names in an ascending order of document distribution value. Later this distribution will be shown as related to the keyword-model performance.

11.1.1 Document Distribution

Document distribution is defined as the ratio between the number of news articles or documents and the number of price swings in a time period. It means that (for a company) if in time period T , the number of times the price went up (in significant amount¹) is N^+_T and the number of times the price went down (in significant amount²) is N^-_T , document distribution or DS (short form of Document/Swing) will be defined as:

$$DS = \frac{D^+_T + D^-_T}{N^+_T + N^-_T} \quad (11.1)$$

where

$$D^+_T + D^-_T = \text{Number of Documents to Analyze} \quad (11.2)$$

and

$$N^+_T + N^-_T = \text{Total Number of Upswings and Downswings} \quad (11.3)$$

Here $D^+_T + D^-_T$ represents the number of documents to analyze: meaning that during that time period T , these many documents (in the form of news articles) were found to be related to the company and were crawled/downloaded for explanation generation purpose.

Now, we present a very interesting observation about the correlation between

¹over a threshold δ^+

²less than a threshold δ^-

Ticker	Name	UP Count	DOWN Count	UP Doc	Down Doc	Total Doc	Doc/Swing
AET	Aetna	32	18	45	44	89	1.78
GE	General Electric	16	10	44	17	61	2.34
LOW	Lowe's Cos	26	20	101	28	129	2.8
AXP	American Express	13	12	66	29	95	3.8
LMT	Lockheed Martin	23	17	123	58	181	4.52
MER	Merrill Lynch	23	18	163	60	223	5.43
BA	Boeing	30	19	134	136	270	5.51
EBAY	eBay Inc.	41	31	184	214	398	5.52
AIG	American Intl Group	23	24	143	140	283	6.02
MOT	Motorola	37	25	342	75	417	6.72
F	Ford Motor	27	30	209	255	464	8.14
AAPL	Apple Computer	43	33	439	240	679	8.93
HPQ	Hewlett-Packard	26	24	300	164	464	9.28
DELL	Dell	21	16	278	110	388	10.48
C	Citigroup	13	10	127	201	328	14.26
INTC	Intel	33	29	501	430	931	15.01
BAC	Bank of America	12	14	134	273	407	15.65
GM	General Motors	28	26	515	481	996	18.44
MSFT	Microsoft	14	13	534	129	663	24.55
IBM	IBM	18	14	565	419	984	30.75

Table 11.1. Details of the dataset for the randomly selected 20 companies. First two columns represents the ticker symbol and the company name. The third and fourth columns respectively represent the number of upswings and downswings of price. The fifth and sixth columns shows the number of articles associated with these swings and thus the seventh column is merely the total number of articles considered for running the keyword model. **The last column shows the document distribution versus the number of price up swing or down swing. This is basically the ratio of total file to analyze and the total number of “UP Count” and “DOWN Count”.**

document distribution and the performance from the keyword-based model. Here, Table 11.1 takes the data from Table 8.1 of Chapter 8, but sorts the entries by Doc/Swing count or **DS** count instead of alphabetical order. Then these data points are plotted in Figure 11.1.

11.1.2 Correlation

Figure 11.2 demonstrates how the **DS** counts of different companies correlate with the keyword model performance.

Basically this graph shows that the variable width (**VW**) approach with the use

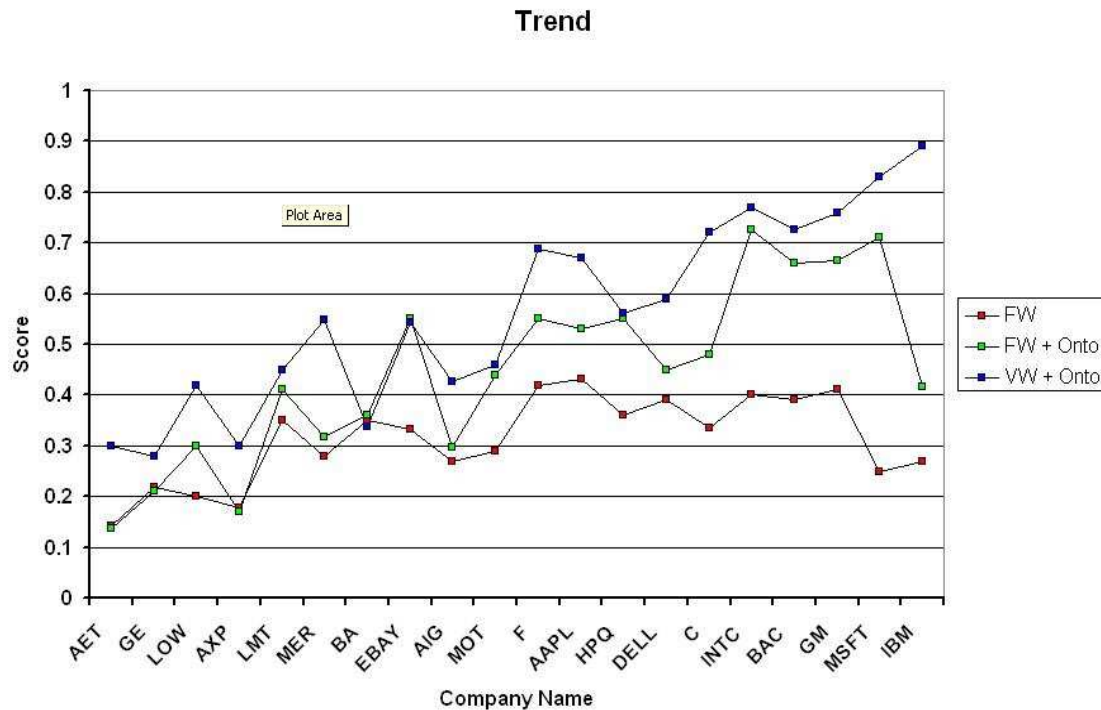


Figure 11.1. The company names are arranged in ascending order of magnitude of **DS** or Document/Swing ratio. **FW** represents fixed window approach. **FW+Onto** shows the results from fixed window approach with the use of ontologies. **VW+Onto** uses variable window and ontologies.

of ontology can bring more relevant segments from the news articles as the number of news articles increases on average. In a simplified term, this means that more data leads to better analysis and better performance score. Other approaches, including **FW** and **FW +Onto** are not exactly aligned with the document distribution, which is intuitively true, as *they always take fixed windows for any pivotal date and those windows may not always catch the right set of documents, which should be analyzed to get the real explanation*. The point to be noted here is that the performance of this analysis also empirically dependent on the choice of proper set of documents, which is also intuitively true. If we can not analyze the right dataset, however better the algorithm is, the quality of the result or the performance will never be good.

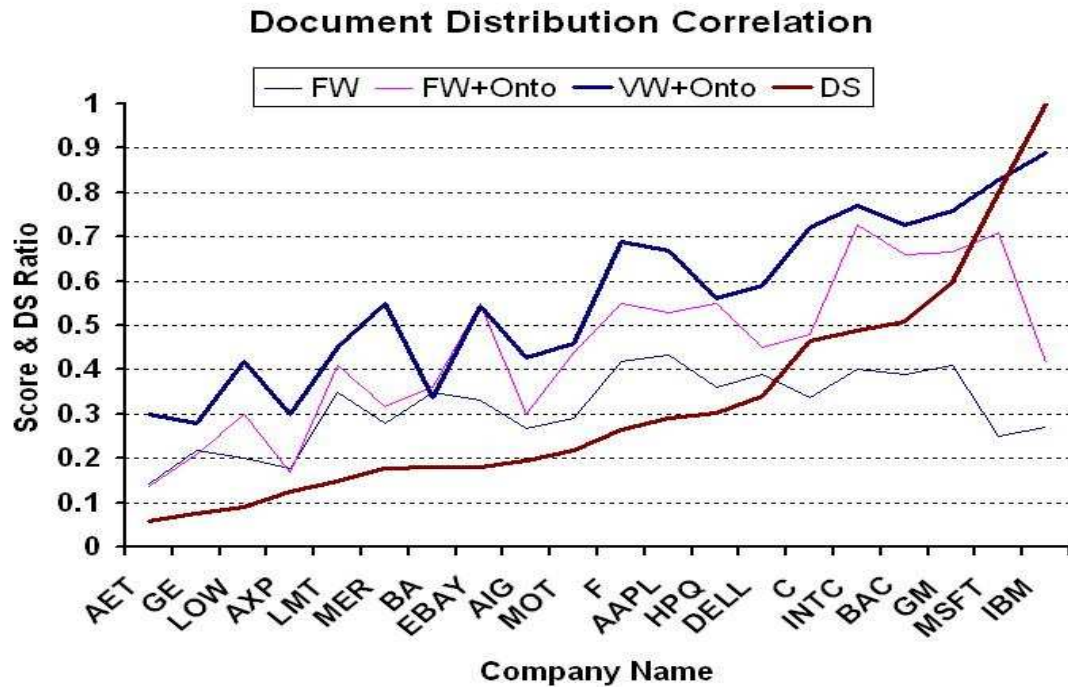


Figure 11.2. Trend comparison of all the keyword models with the **DS** count value. This graph shows that the **VW + Onto** model correlates with the **DS** count most closely. This indicates that the variable width model with the use of ontology can bring more relevant portions of the news articles as the number of news articles increases on average. And more data leads to better analysis and better performance score.

11.2 Other Observations

There are other interesting observations about how one company's up or down swing of price can help or affect another company's price. Two such examples of opposite nature are shown here.

- The first set consists of Figures 11.3, 11.4, and Table 11.2. It is shown that in a flourishing business sector, a bad news of a competitor can help increase a company's stock price.
- Similarly the second set consists of Figures 11.5, 11.6, and Table 11.3. Here it is shown that in a shaky market, where not many companies are doing well, a good news for a certain company can boost the price of even a rival



Figure 11.3. Price graph for HPQ(Hewlett Packard)

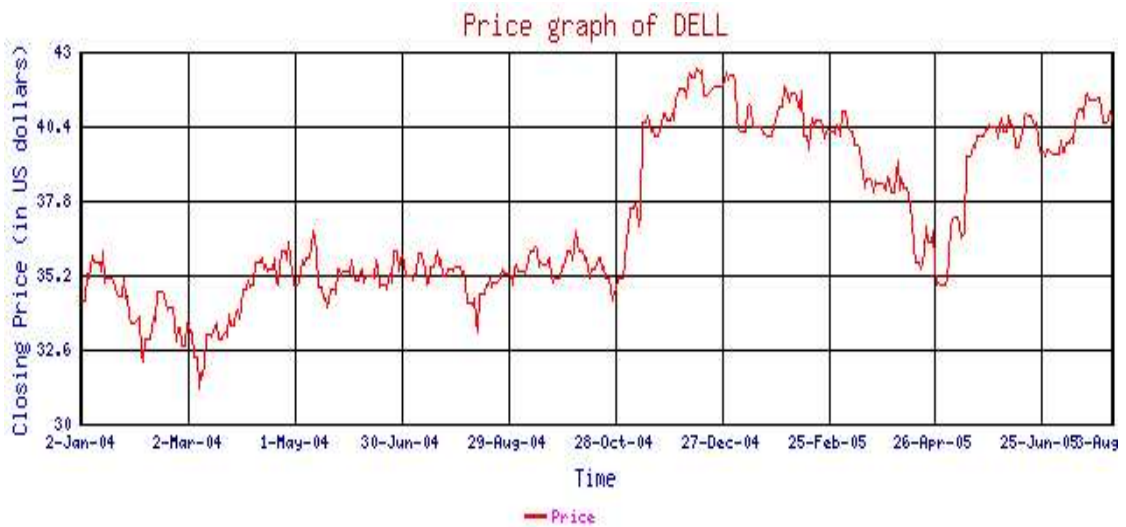


Figure 11.4. Price graph for DELL(Dell)

company. In this scenario a good news of any company actually influences the whole sector, and as people's confidence on the sector increases, prices of other companies, even if they are competitors can be affected.

Ticker	Date	Price Swing	Real Cause/Fact/Analysis	Keywords Generated
HPQ (Hewlett Packard)	Feb 8-10, 2005	UP	Dell's fourth-quarter profit edged expectations, but after-hours investors were shaken by revenue that was merely in line, as well as a top-line forecast below the current Street consensus.	forecast, shaken, net, stay, billion

Table 11.2. One instance from the keyword model for Hewlett Packard is shown here. As Dell and Hewlett Packard are competitors in a flourishing business sector (computer hardware), reports of a bad news of rival (Dell) acted as a boon to Hewlett Packard. Simply, **rival's bad news is good news in a flourishing sector**



Figure 11.5. Price graph for F(Ford Motors)

11.3 Conclusion

Though keyword model is the primary method we used here, we also tried a new model of explanation generation, which is based on sentence analysis. As we have described briefly in chapter 6, sentence model is worth trying for showing more presentable results. In the next chapter we will see some interesting results from this model and we will conclude afterwards.



Figure 11.6. Price graph for GM(General Motors)

Ticker	Date	Price Swing	Real Cause/Fact/Analysis	Keywords Generated
F (Ford)	April 29 - May 4, 2005	UP	Standard & Poor downgraded the credit ratings of automakers like Ford and GM to junk status – however billionaire Kirk Kerkorian offered to buy big chunks of GM shares. Microsoft chairman Bill Gates and Ford chairman and chief executive Bill Ford Jr. said having high-definition screens in vehicles, speech recognition, cameras, digital calendars and navigation equipment with directions and road conditions will set car companies apart from their competitors.	Thursday, interest, bill, gates, kirk, billionaire, kerkorian
GM (General Motors)	April 29 - May 4, 2005	UP	Standard & Poor downgraded the credit ratings of automakers like Ford and GM to junk status – however billionaire Kirk Kerkorian offered to buy big chunks of GM shares.	kirk, likely, billionaire, kerkorian, expectations

Table 11.3. Two instances from the keyword model for Ford and General Motors, competitors in the auto industry. But the industry is recently not doing well. In this shaky market, if something good happens to any company (here GM), then it influences others as well, even if they are competitors. Here as billionaire Kirk Kerkorian offered to buy a big chunk of General Motors share, not only did General Motors share price go up, but so did it Ford's share. This is a very interesting example of the fact that the market felt more confident in the whole sector as one of the player's (GM's) shares went up due to a good news.

Chapter 12

Conclusion and Sentence-based Explanation Model

‘Finally, I will not become any dumber.’

- **Paul Erdős**

(Hungarian Gypsy Mathematician)

(1913-1996)

In this chapter we conclude our work and show some of the preliminary results of our ongoing work to find sentence-based explanation. In this model we go even further and deeper to assist users with an explanation which is presented as a sentence or a set of sentences. At first look it seems that this model should not be too different from the keyword-based model (Chapters 6, 8, 10).

12.1 Why We Need a Separate Model

Let us discuss why we need a completely separate model for sentence-based explanation [42].

- If we can extract explanatory keywords by applying the entropy model or

Kullback-Liebler divergence on the keywords, it should be possible to generate explanatory sentences if we can feed the same entropy model with sentences. However, this does not work as the basic granularity of any news article is not a sentence, it is a word. The probability of finding the exact same sentence in two articles is almost nil. On the contrary, as words are the basic granular or atomic identity of an article, paragraph or a sentence, the same word can occur several times in a single paragraph or multiple paragraphs. So applying the entropy model or the Kullback-Liebler divergence will not produce any result.

- Similarly we can think of extending the keyword-based explanation model to generate the sentence from an article. If we can associate a simple mapping from the words to the article and to the sentences inside an article, then once we get the word, we can refer to the article and the sentence from which it is coming. However, first of all, we lose the mapping between a word and a sentence immediately when we just take different probability measures of the word (by word count). It is not possible to associate each word with the sentence reference while we are measuring the probabilities and entropies.
- Thirdly, we can think of an ad-hoc technique where we may not need the aforesaid reference. After getting the keywords, we can just search through the article set (in W_A) to find the sentence from which it can possibly come. This approach, however, is too simplistic and can not guarantee any performance if implemented. However, in practice, implementing this simplistic approach may not be that simple. During the keyword generation and keywords probability measures, if we rely upon stemming [111], then the actual form of the word gets lost. Therefore, during the last step, if we also have to stem each word in a candidate sentence, the chances of getting multiple sentences for the word increases. In real cases, we do stem the words and so the real form of the word is already lost.

Due to all these fundamental difficulties, we realized that sentence-based explanation generation is a very different problem than the keyword-based approach. In the next section we describe the architecture and theory behind this model.

12.2 Sentence-based Model Architecture

Our sentence-based model is a complex model and is described in Figure 12.1. The main difference between the keyword-model and the sentence-model is that in the sentence model we do not consider two different windows of documents. We only consider one window which is representative of the pivotal date according to the price trend. This is the same window as the variable-window in the keyword-model. As there is only one window or one set of documents analyzed, there is no concept of entropy or difference between the two sets.

In the sentence-based model, we only consider the “Window A” as defined in chapter 6, and no “Window B”. Therefore

$$W_A^v = -(d_A - d) \text{ where } slope(d) \neq slope(d_{(a-1)}), \forall a, A = max(a), \text{ and } d > d_A \quad (12.1)$$

And so the only document set analysed is the document or news collection gathered during the pivotal date and during the d_A days just before the pivotal date.

$$C_A^v = C(d) + C(d_A - d) \quad (12.2)$$

12.3 Language Model

Language models are recently gaining popularity among IR researchers. We extracted necessary keywords for generating the language models by the steps as follows. These steps ensure that these language models can be learned from the previous articles (training articles).

1. **Selecting the Time Frame:** First we select a time frame for the learning period. In our case this duration was from Jan 1, 2002 to Dec 31, 2004. Let’s say this price duration is T .
2. **Collecting the News Articles:** We collected whatever news articles were available for 200 US companies. As we were building the models for price

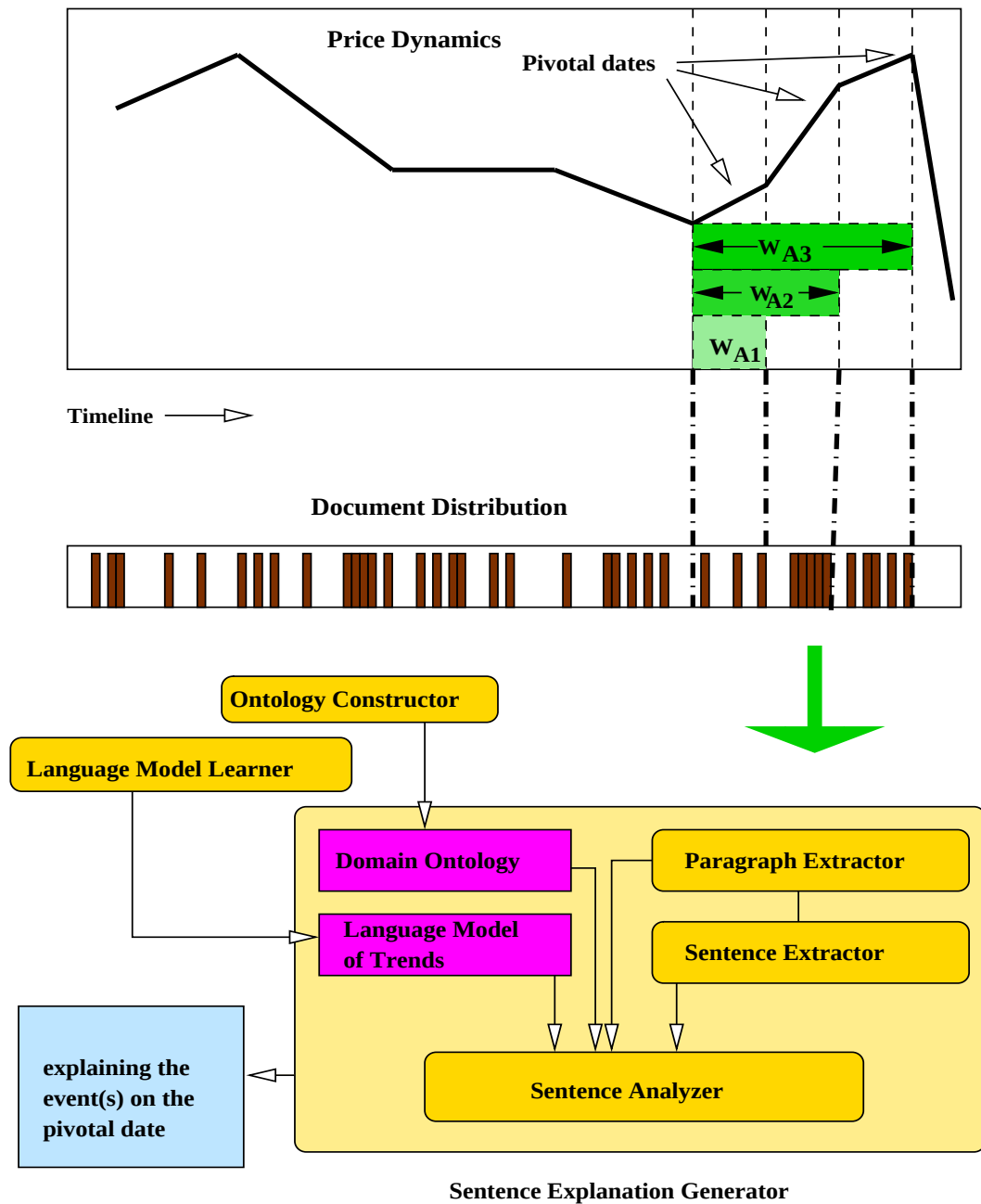


Figure 12.1. A simple way to show the analysis procedure for explanatory sentence generation. The date for which we are trying to find an explanation is called the pivotal date. Only the documents during or before the pivotal date with same price trend as the pivotal date are considered for the analysis.

movements, we kept the models independent of the companies and dependent only on price movements. Let us assume this document set is $D(T)$, which consists of documents from all companies.

$$D(T) = \bigcup (D_1(T), D_2(T), \dots, D_N(T)) \quad (12.3)$$

where $N = 200$, the number of companies, and $D_i(T)$ is the document (news article) set for i^{th} company for the time duration T .

3. **Separating the News Articles:** We separate the articles according to the major price swings and put them in two separate bins, **UP Bin** and **DOWN Bin**. As the number of articles grew too large, we only took the price swings which exceeded 5%. Let us assume these bins are $D^{UP}(T)$ and $D^{DOWN}(T)$
4. **Tagging the Paragraphs:** This part was tedious work. We had to tag part of these articles very quickly to curb out the unnecessary part and to reduce the amount of sentences/words used to make these models. The tagged portions represent the possible effect of/fact of/reason for the price movement.
5. **POS Tagging:** We used Brill's POS tagger to tag all the important portions (paragraphs) of these articles, which might possibly affect the price swings during those time periods. This way we can separate the nouns, verbs, or articles and only take the necessary ones which can possibly affect prices. We took mainly verbs and some nouns which could possibly be responsible. This part is also manual and took a while to build.
6. **Building the Learned Model:** After we got the words responsible for the price swings, we created the language model with them. The models also include frequencies of each word. We specify the models as M^+_{pt} for UP swing and M^-_{pt} for DOWN swing.

Company	Date	Sentence
MSFT	July 1, 2005	IBM Corp. will get \$775 million in cash and \$75 million worth of software from Microsoft Corp. to settle claims ...
C	July 11, 2005	Merrill Lynch cut its second-quarter earnings estimate for Citigroup, Inc. to \$1.04 from \$1.09 a share and slashed its fiscal 2005 earnings estimate...
IBM	April 15, 2005	1. IBM fell \$6.94, to close at \$76.70, as investors reacted to the company's surprise earnings announcement. ... 2. ...Big Blue's report came out, some analysts speculated there could be weakness in IBM's services business...
DELL	May 10, 2005	Billionaire Dell Invests In Linux Giant Red Hat...

Table 12.1. Explanatory sentences for companies for a specified date

12.4 Explanatory Sentences

12.4.1 Language Model

Let us assume that C^v_A denotes the document set inside the W^v_A window (variable window) for a price swing. Let us assume that the price swing was UP and we will derive the equations for it (the derivation for DOWN swing will be similar). Let us assume D denotes the number of documents in the set C^v_A .

$$D = |C^v_A| \quad (12.4)$$

Let suppose that document i has S_i sentences and so the total number of sentences $\mathcal{S}$ in the set is

$$\mathcal{S} = \sum_i^D |S_i| \quad (12.5)$$

With this set up, we calculate the probability distribution of the sentences given the language model and then calculate it for all the sentences. If we denote this score as L_j for a sentence S_j then

$$L_j = Prob(S_j | M^+_{pt}), \forall j \in \mathcal{S} \quad (12.6)$$

12.4.2 Ontology-based Relevance Model

We used the ontologies built as shown in Chapter 9 and give scores to all the sentences. We then take R_j as the score from this relevance model,

$$R_j = Prob(S_j|O^C), \forall j \in \mathcal{S} \quad (12.7)$$

where O^C denotes the ontology for the company C , which is basically the word-set coming from their ontologies.

12.4.3 Mixture Model

We use a mixture model for finding the explanatory sentences. We calculate the following score for all the sentences. If this final score is denoted as F_j then

$$S_{expl} = argmax_j(F_j|F_j = w_L \times L_j + w_R \times R_j) \quad (12.8)$$

where w_L and w_R are the weights to control the effect of the ontologies for sentence based relevance.

12.5 Results

We used the above method with some minor variations and got decent results from several different companies [53]. We also tried to use news **HeadLine** to extract the themes of the stories (exploiting the fact that **HeadLines** often represents a one-liner for the main theme) by matching main keywords in the **HeadLine** and in each sentence of the story [40]. The results are shown in Table 12.1.

12.6 Conclusion

We studied the keyword-based model extensively and show that it can work as an effective tool to find keyword-based explanation for an event with which we can associate a time-frame and for which we can collect some relevant documents. We decided to use market domain as there are two distinct characteristics: price and related market events.

Firstly, we demonstrated that price and related market events are well-correlated. In that context we picked up a few artificial markets and betting markets, and we studied them extensively.

Secondly, we extracted important text portions, and removed redundant portions of a Web-page (news article) to get important text-based data.

Thirdly, we extracted the temporal information and publishing time of these news articles with high accuracies and so could align these news articles with time-frame very well.

We analyzed these articles according to the price dynamics and extracted useful keywords (by using entropy-based model and KL divergence) which can explain the price dynamics. In the real stock market, as these techniques did not work out well, we planned to extract more relevant information by using domain ontologies. We built domain ontologies for all companies we were analyzing and applied them to get improved results. We used term-relationship and a few other rule-based extractor methods to collect terms to build these ontologies. We also introduced a variation of the keyword-model which is variable window approach, and it worked out better than using just ontology over the basic fixed window model.

At the end, we also looked into other possibilities and a learning-based approach to extract useful explanation. For this, we introduced sentence-based model. Though it is still in the nascent stage, sentence-based model is a promising model to find explanatory sentences when we can learn the language models from past articles. Preliminary results show better success with an ontology model than without it. We hope that in future this model will be studied and analyzed more extensively.

Bibliography

- [1] <http://www.google.com>.
- [2] <http://www.yahoo.com>.
- [3] <http://www.altavista.com>.
- [4] <http://www.askjeeves.com>.
- [5] <http://www.vivisimo.com>.
- [6] <http://www.xml.org>.
- [7] <http://misshoover.si.umich.edu/~zzheng/qa-new/>.
- [8] <http://www.ionaut.com:8400/>.
- [9] http://hal.comp.nus.edu.sg/cgi-bin/smadellz/lamp_query.pl.
- [10] <http://ciir.cs.umass.edu/~reu2/>.
- [11] <http://www.ai.mit.edu/projects/infolab/>.
- [12] <http://www.wondir.com/wondir/jsp/index.jsp>.
- [13] <http://forbes.com>.
- [14] S. Abiteboul. Querying semi-structured data. In *Proceedings of International Conference on Database Theory*, pages 1–18, Delphi, Greece, 1997. Springer.

- [15] Charu C. Aggarwal, Fatima Al-Garawi, and Philip S. Yu. Intelligent crawling on the world wide web with arbitrary predicates. In *Proceedings of WWW10*, pages 96–105, 2001.
- [16] J. F. Allen. Towards a general theory of action and time. In *Artificial Intelligence*, pages 123–154, 1984.
- [17] J. F. Allen. Natural language understanding: Discourse structure, tense and aspect. In *Addison-Wesley Chapter 16:5*, pages 517–533, 1995.
- [18] V. Almeida, D. Menasce, R. Reidi, F. Peligrinelli, R. Fonseca, and W. M. Jr. Analyzing web robots and their impact on caching. In *In Proceedings of the 6th Web Caching and Content Delivery Workshop*, 2001.
- [19] J. L. Ambite, N. Ashish, G. Barish, C. A. Knoblock, S. Minton, P. J. Modi, I. Muslea, A. Philpot, and S. Tejada. Ariadne: a system for constructing mediators for internet sources. In *SIGMOD*, pages 561–563, 1998.
- [20] Giuseppe Attardi, Antonio Gullí, and Fabrizio Sebastiani. Automatic Web page categorization by link and context analysis. In Chris Hutchison and Gaetano Lanzarone, editors, *Proceedings of THAI-99, European Symposium on Telematics, Hypermedia and Artificial Intelligence*, pages 105–119, Varese, IT, 1999.
- [21] P. Atzeni, G. Mecca, and P. Merialdo. Semistructured and structured data in the web: Going back and forth. In *Workshop on Management of Semistructured Data*, 1997.
- [22] Ziv Bar-Yossef and Sridhar Rajagopalan. Template detection via data mining and its applications. In *Proceedings of WWW 2002*, pages 580–591, 2002.
- [23] Klaus B. Beckmann and Martin Werding. Markets and the use of knowledge: Testing the hayek hypothesis in experimental stock markets. Technical Report Discussion paper LSL 94-02, University of Passau, 1994.
- [24] Sanjiv K. Bhatia. Selection of search terms based on user profile. In *Proceedings of the ACM/SIGAPP Symposium on Applied computing*, pages 224–233, 1992.

- [25] M. Bianchini, M. Gori, and F. Scarselli. Pagerank: A circuital analysis, 2002.
- [26] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
- [27] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Block based web search. In *27th Annual International ACM SIGIR Conference*, pages 456–463, 2004.
- [28] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1623–1640, 1999.
- [29] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The tsimmi project: integration of heterogeneous information sources. In *Proceedings of the 10th meeting of Information Processing Society of Japan*, pages 7–18, 1994.
- [30] B. Chidlovskii, J. Ragetli, and M. de Rijke. Wrapper generation via grammar induction. In *Machine Learning: ECML 2000, 11th European Conference on Machine Learning, Barcelona, Catalonia, Spain, May 31 - June 2, 2000, Proceedings*, volume 1810, pages 96–108. Springer, Berlin, 2000.
- [31] Junghoo Cho and Hector Garcia-Molina. Parallel crawlers. In *In Proceedings of the 11th World Wide Web conference (WWW11)*, 2002.
- [32] W. W. Cohen. A web-based information system that reasons with structured collections of text. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents’98)*, pages 400–407, New York, 9–13, 1998. ACM Press.
- [33] World Wide Web Consortium. World wide web consortium hypertext markup language.
- [34] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic

- knowledge from the world wide web. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 509–516, Madison, US, 1998. AAAI Press, Menlo Park, US.
- [35] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 109–118, 2001.
 - [36] P. M. E. De Bra and R. D. J. Post. Information retrieval in the world-wide web: Making client-based searching feasible. *Computer Networks and ISDN Systems*, 27(2):183–192, 1994.
 - [37] Luis M. de Campos, Juan M. Fernandez, and Juan F. Huete. Query expansion in information retrieval systems using a bayesian network-based thesaurus. In *Proceedings of 1st UAI*, pages 53–60, New York, NY, 1985. Elsevier Science Publishing Comapny, Inc.
 - [38] Luis M. de Campos, Juan M. Fernandez, and Juan F. Huete. Building bayesian network-based information retrieval systems. In *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, page 543, 2000.
 - [39] Luis M. de Campos, Juan M. Fernandez-Luna, and Juan F. Huete. A layered bayesian network model for document retrieval. In *Proceedings of the 24th BCS-IRSG European Colloquium on IR Research*, pages 169–182, 2002.
 - [40] Sandip Debnath and C Lee Giles. A learning based model for headline extraction of news articles to find explanatory sentences for events. In *the proceedings of the upcoming Knowledge Capture conference*, 2005.
 - [41] Sandip Debnath and C Lee Giles. Extracting news metadata: A learning-based approach. In *submission for IEEE TKDE Journal*, 2006.
 - [42] Sandip Debnath and C Lee Giles. A sentence-based explanation model. In *submission for IEEE TKDE Journal*, 2006.

- [43] Sandip Debnath, Prasenjit Mitra, , and C Lee Giles. Identifying primary content blocks from web pages. Technical report, The Pennsylvania State University, 2004.
- [44] Sandip Debnath, Prasenjit Mitra, and C. Lee Giles. Automatic extraction of informative blocks from webpages. In *proceedings of the Special Track on Web Technologies and Applications in the ACM Symposium of Applied Computing*, pages 1722–1726, 2005.
- [45] Sandip Debnath, Prasenjit Mitra, and C. Lee Giles. Finding base time-line of a news article. In *AAAI FLAIRS 2005 Conference*, pages 142–147, 2005.
- [46] Sandip Debnath, Prasenjit Mitra, and C. Lee Giles. Identifying content blocks from web documents. In *In proceedings of the 15th International Symposium on Methodologies for Intelligent Systems*, pages 285–293, 2005.
- [47] Sandip Debnath, Prasenjit Mitra, Nirmal Pal, and C. Lee Giles. Automatic identification of informative sections of web pages. In *IEEE Transactions on Knowledge and Data Engineering*, volume 17(9), pages 1233–1246, 2005.
- [48] Sandip Debnath, Tracy Mullen, Arun Upneja, and C Lee Giles. Knowledge discovery in web-directories: Finding term-relations to build a business ontology. In *the proceedings of the upcoming ACM EC-WEB*, 2005.
- [49] Sandip Debnath, David Pennock, and C Lee Giles. A keyword-based explanation model. In *submission for IEEE TKDE Journal*, 2006.
- [50] Sandip Debnath, David M. Pennock, C. Lee Giles, and Steve Lawrence. Information incorporation in online in-game sports betting markets. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 258–259, San Diego, CA, USA, 2003. ACM Press New York, NY, USA.
- [51] Sandip Debnath, David M. Pennock, Steve Lawrence, Eric J. Glover, and C. Lee Giles. Characterizing efficiency and information incorporation in sports betting markets. In *Proceedings of the Ninth Research Symposium on Emerging Electronic Markets*, pages 39–52, University of Applied Sciences Basel (FHBB), Institute for Business Economics, Basel, Switzerland, 2002.

- [52] Sandip Debnath, Sandip Sen, and Brent Blackstock. Lawbot: A multiagent system for legal research. In *IEEE Internet Computing Journal*, volume 4(6), pages 32–37. IEEE, 2000.
- [53] Sandip Debnath, Arun Upneja, and C Lee Giles. Learning term-relationships for ontology construction: Creating business ontologies for event explanation. In *the proceedings of the upcoming Knowledge Capture conference*, 2005.
- [54] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, pages 527–534, Cairo, Egypt, 10–14 September 2000.
- [55] B. Dorr and M. B. Olsen. Driving verbal and coompositional lexical aspect for nlp applications. In *the proceedings of ACL*, pages 151–158, 1997.
- [56] Efthimis N. Efthimiadis. A user-centred evaluation of ranking algorithms for interactive query expansion. In *Proceedings of the 16th ACM SIGIR*, pages 146–159, 1993.
- [57] L. Ferro, I. Mani, B. Sundheim, and G. Wilson. Tides temporal annotation guidelines draft - version 1.02. In *MITRE Technical report*, 2001.
- [58] Robert Forsythe and Russell Lundholm. Information aggregation in an experimental market. *Econometrica*, 58(2):309–347, 1990.
- [59] Robert Forsythe, Thomas A. Rietz, and Thomas W. Ross. Wishes, expectations, and actions: A survey on price formation in election stock markets. *Journal of Economic Behavior and Organization*, 39:83–110, 1999.
- [60] Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *Information Systems*, 9(3):223–248, 1991.
- [61] Norbert Fuhr and Chris Buckley. Optimizing document indexing and search term weighting based on probabilistic models. In *Text REtrieval Conference*, pages 89–100, 1992.

- [62] Robert Fung and Brendan Del Favero. Applying bayesian networks to information retrieval. In *Communications of the ACM*, volume 38(3), pages 42–ff, 1995.
- [63] John M. Gandar, William H. Dare, Craig R. Brown, and Richard A. Zuber. Informed traders and price variations in the betting market for professional basketball games. *Journal of Finance*, LIII(1):385–401, 1998.
- [64] Sanford J. Grossman. An introduction to the theory of rational expectations under asymmetric information. *Review of Economic Studies*, 48(4):541–559, 1981.
- [65] David Haines and W. Bruce Croft. Relevance feedback and inference networks. In *Proceedings of the 16th ACM SIGIR*, pages 2–11, 1993.
- [66] Robin D. Hanson. Could gambling save science? encouraging an honest consensus. *Social Epistemology*, 9(1):3–33, 1995.
- [67] Donna Harman. Towards interactive query expansion. In *Proceedings of the 11th ACM SIGIR*, pages 321–331, 1988.
- [68] Donna Harman. Ranking algorithms. In *Information Retrieval: Data Structures and Algorithms*, pages 363–392. Englewood Cliffs: Prentice Hall, 1992.
- [69] Donna Harman. Relevance feedback and other query modification techniques. In *Information Retrieval: Data Structures and Algorithms*, pages 241–263. Englewood Cliffs: Prentice Hall, 1992.
- [70] Donna Harman. Relevance feedback revisited. In *Proceedings of the 15th ACM SIGIR*, pages 1–10, 1992.
- [71] Wu Harry and Gerard Salton. A comparison of search term weighting: term relevance vs. inverse document frequency. In *Proceedings of the 4th ACM SIGIR*, pages 30–39, 1981.
- [72] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data mining, Inference, and Prediction*. 2003.

- [73] C. Hsu. Initial results on wrapping semistructured web pages with finite-state transducers and contextual rules. In *AAAI-98 Workshop on AI and Information Integration*, pages 66–73. AAAI Press, 1998.
- [74] C.W. Hsu, C.C. Chang, and C.J. Lin. A practical guide to support vector classification. In *A Library of Support vector Machines*.
- [75] <http://finance.yahoo.com>, 2003.
- [76] Jens Carsten Jackwerth and Mark Rubinstein. Recovering probability distributions from options prices. *Journal of Finance*, 51(5):1611–1631, December 1996.
- [77] George A. Kelly. A mathematical approach to psychology. In *B. Maher, Ed. Clinical Psychology and Personality: The Selected Papers of George Kelly*, pages 94–112. John Wiley and Sons, 1969.
- [78] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. In *Proceedings of the AAAI Spring Symposium: Information Gathering from Heterogeneous Distributed Environments*, pages 85–91, 1995.
- [79] D.B. Koen and W. Bender. Time frames: Temporal augmentation of the news. In *IBM Systems Journal*, volume 39, pages 597–616, 2000.
- [80] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000.
- [81] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–737, 1997.
- [82] A. Lascarides and N. Asher. Temporal relations, discourse structure, and commonsense entailment. In *Linguistics and Philosophy*, pages 437–494, 1993.
- [83] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems -*

- Special Issue on Networked Information Discovery and Retrieval*, 5(2):121–143, 1995.
- [84] S. Lin and J. Ho. Discovering informative content blocks from web documents. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 588–593, 2002.
 - [85] Bing Liu, Kaidi Zhao, and Lan Yi. Eliminating noisy information in web pages for data mining. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 296–305, 2003.
 - [86] Robert E. Lucas. Expectations and the neutrality of money. *Journal of Economic Theory*, 4(2):103–24, 1972.
 - [87] I. Mani and G. Wilson. Robust temporal processing of news. In *the proceedings of ACL*, pages 69–76, 2000.
 - [88] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification, 1998.
 - [89] W. Meng, K. Liu, C. Yu, W. Wu, and N. Rishe. Estimating the usefulness of search engines. In *15th International Conference on Data engineering, ICDE*, Sydney, Australia, 1999.
 - [90] Rada Mihalcea and Dan I. Moldovan. Document indexing using named entities, 2001.
 - [91] MITRE. Alembic workbench - <http://www.mitre.org/tech/alembic-workbench/>.
 - [92] Stefano Mizzaro. Relevance: the whole history. In *Journal of the American Society for Information Science*, volume 48, pages 810–832, 1997.
 - [93] I. Muslea, S. Minton, and C. A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001.

- [94] R. Nallapati and J. Allan. Capturing term dependencies using a language model based in sentence tree. In *Proceedings of CIKM 2002*, 2002.
- [95] Gonzalo Navarro and Ricardo Baeza-Yates. A language for queries on structure and contents of textual databases. In *Proceedings of ACM SIGIR'95*, pages 93–101, 1995.
- [96] Berthier Ribeiro Neto and Richard Muntz. A belief network model for ir. In *Proceedings of the 19th ACM SIGIR*, pages 253–260, 1996.
- [97] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
- [98] Gautam Pant, Padmini Srinivasan, and Filippo Menczer. Crawling the web.
- [99] R. J. Passanneau. A computational model of the semantics of tense and aspect. In *Computational Linguistics*, pages 44–60, 1988.
- [100] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1988.
- [101] Helen J. Peat and Peter Willett. The limitations of term co-occurrence data for query expansion in document retrieval systems. In *Journal of the American Society for Information Science*, volume 42(5), pages 378–383, 1991.
- [102] David Pennock, Sandip Debnath, Eric Glover, and C. Lee Giles. Modelling information incorporation in markets, with application to detecting and explaining events. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 405–413, San Francisco, CA, 2002. Morgan Kaufmann Publishers.
- [103] David M. Pennock, Steve Lawrence, C. Lee Giles, and Finn Arup Nielsen. The power of play: Efficiency and forecast accuracy in web market games. Technical Report 2000-168, NEC Research Institute, 2000.

- [104] David M. Pennock, Steve Lawrence, C. Lee Giles, and Finn Arup Nielsen. The real power of artificial markets. *Science*, 291:987–988, February 9 2001.
- [105] David M. Pennock, Steve Lawrence, Finn Arup Nielsen, and C. Lee Giles. Extracting collective probabilistic forecasts from web games. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 174–183, 2001.
- [106] Ulrich Pfeifer, Norbert Fuhr, and Tung Huynh. Searching structured documents with the enhanced retrieval functionality of freewais-sf and sfgate. In *Computer Networks and ISDN Systems*, volume 27(6), pages 1027–1036, 1995.
- [107] Justin Picard and Rolf Haenni. Modeling information retrieval with probabilistic argumentation systems. In *20th Annual BCS-IRSG Colloquium on IR*, 1998.
- [108] C. R. Plott, J. Wit, and W. C. Yang. Parimutuel betting markets as information aggregation devices: Experimental results. Technical Report Social Science Working Paper 986, California Institute of Technology, April 1997.
- [109] Charles R. Plott and Shyam Sunder. Efficiency of experimental security markets with insider information: An application of rational-expectations models. *Journal of Political Economy*, 90(4):663–98, 1982.
- [110] Charles R. Plott and Shyam Sunder. Rational expectations and the aggregation of diverse information in laboratory security markets. *Econometrica*, 56(5):1085–1118, 1988.
- [111] M. F. Porter. An algorithm for suffix stripping. In *Program*, volume 14(3), pages 130–137, 1980.
- [112] S. Raghavan and H. Garcia-Molina. Complex queries over web repositories, 2003.
- [113] Lakshmish Ramaswamy, Arun Iyengar, Ling Liu, and Fred Douglass. Automatic detection of fragments in dynamically generated web pages. In *Thirteenth World Wide Web conference*, pages 443–454, 2004.

- [114] Lakshmish Ramaswamy, Arun Iyengar, Ling Liu, and Fred Douglass. Automatic fragment detection in dynamic web pages and its impact on caching. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):1–16, 2005.
- [115] H. Reichenbach. The tenses of verb. In *Elements of Symbolic Logic*, pages 287–298, 1947.
- [116] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.
- [117] W. Sack. The questioning news system. In *Technical Report presented at the MIT media Library*, 1997.
- [118] Gerard Salton. *Automatic Information Organization and Retrieval*. McGraw-Hill, 1968.
- [119] Gerard Salton and C. Buckley. Improving retrieval performance by relevance feedback. In *Journal of the American Society for Information Science*, volume 41, pages 288–297, 1990.
- [120] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc. New York, NY, USA, 1986.
- [121] C. Schmidt and A. Werwatz. How accurate do markets predict the outcome of an event? the euro 2000 soccer championships experiment. In *Max Planck Institute for Research into Economic Systems working paper*, 2002.
- [122] B. Schölkopf, A. Smola, R. Williamson, and P. L. Bartlett. New support vector algorithms. neural computation. In *Neural Computation 12*, pages 1207–1245, 2000.
- [123] I. R. Silva. Bayesian networks for information retrieval systems. In *PhD thesis, Universidade Federal de Minas Gerais*, 2000.
- [124] A. Smeaton and C. J. van Rijsbergen. The retrieval effects of query expansion on a feedback document retrieval system. In *Computer Journal*, volume 26(3), pages 239–246, 1983.

- [125] Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma. Learning block importance models for web pages. In *Thirteenth World Wide Web conference*, pages 203–211, 2004.
- [126] Ellen Spertus and Lynn Andrea Stein. Just-in-time databases and the world-wide web. In *CIKM*, pages 30–37, 1998.
- [127] Ellen Spertus and Lynn Andrea Stein. Squeal: a structured query language for the Web. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):95–103, 2000.
- [128] P. Srinivasan, J. Mitchell, O. Bodenreider, G. Pant, and F. Menczer. Web crawling agents for retrieving biomedical information. In *Proc. Int. Workshop on Agents in Bioinformatics (NETTAB-02)*, 2002.
- [129] Richard H. Thaler and William T. Ziemba. Anomalies: Parimutuel betting markets: Racetracks and lotteries. *Journal of Economic Perspectives*, 2(2):161–174, 1988.
- [130] H. Turtle and W. B. Croft. Inference networks for document retrieval. In *Proceedings of the 13th ACM SIGIR*, pages 1–24, 1990.
- [131] Mike Uschold and Austin Tate. Putting ontologies to use. In *The Knowledge Engineering Review*, volume 13(1), March 1998.
- [132] Robert L. Winkler and Allan H. Murphy. Good probability assessors. *J. Applied Meteorology*, 7:751–758, 1968.
- [133] Lan Yi, Bing Liu, and Xiaoli Li. Visualizing web site comparisons. In *Proceedings of the eleventh international conference on World Wide Web*, pages 693–703, 2002.
- [134] Dell Zhang and Wee Sun Lee. Web taxonomy integration using support vector machines. In *Proceedings of WWW 2004*, 2004.

Vita

Sandip Debnath

HIGHLIGHTS

- 3+ years of IT job experience including jobs at Wipro.com, internships in Webel/Philips, NEC Lab, Overture/Yahoo Lab, Smeal Business School Lab
- Main designer and architect of Smealsearch (departmental search engine of Smeal College of Business)
- Current Erdos # 5 (P.Erdos $\rightarrow$ F.R.K.Chung $\rightarrow$ R.E.Tarzan $\rightarrow$ G.W.Flake $\rightarrow$ D.M.Pennock $\rightarrow$ S. Debnath)
- Distinction/Honors in M.Tech (Computer Science)
- First Class with Honors in B.E.T.C.E
- Stood 3rd (Medical) and 43rd (Engineering) in State-wise Joint Entrance Examinations (JEE 1990) among approximately 0.6 million students
- Diploma in Indian classical music (Hawaiian guitar)
- Language: Bengali, English, Sanskrit, Hindi, Gujarati, Spanish and Russian (learning)

EDUCATION

- **PhD** (Computer Science) The Pennsylvania State University, PA, 2005
20+ publications, GPA: 3.82
- **M.S.** (Computer Science) University of Tulsa, OK, Jan 1999-Dec 2000
- **M.Tech** (Computer Science) Indian Statistical Institute, India, Aug 1996-Aug 1998
Topper with First Class and Distinction
- **B.E.** (Electronics and Telecommunications Engineering), Jadavpur University, India, Aug 1990-Aug 1994
First Class with Honors