SIDEFFECTIVE - SYSTEM TO MINE PATIENT REVIEWS: SENTIMENT ANALYSIS

BY DEEPAK YALAMANCHI

A thesis submitted to the
Graduate School—New Brunswick
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Master of Science
Graduate Program in Computer Science

Written under the direction of
Prof. Tomasz Imielinski

and approved by

__________________________

__________________________

__________________________

__________________________

New Brunswick, New Jersey
May, 2011
Abstract of the Thesis

Sideffective - System to Mine Patient Reviews: Sentiment Analysis

by Deepak Yalamanchi

Thesis Director: Prof. Tomasz Imielinski

Sideffective is the system to crawl, rank and analyze patient testimonials about side effects from common medications. Since the wealth of any mining model is the Data corpus, the data collection phase involved extensive crawling of massive medical websites comprised of user forums from the internet. Subsequently, the raw files were subjected to certain site-specific parsing routines, yielding outputs conforming to a well-defined data model. Currently, the system holds close to 400,000 user testimonials pertaining to more than 2500 drugs/medicines. Sideeffective aims at gathering and aggregating this wealth of information, build useful associations and present interesting observations and numeric validations, all in a user-friendly interface. The important issues that we have tried to tackle are: Extracting side effects without relying on pre-built lists, aggregating distribution of different side effect for a give drug, site-specific search, ranking and determining the negativity of reviews. The system has been jointly built by Deepak Yalamanchi and Sangeetha Rajagopalan under the guidance of Prof. Tomasz Imielinski.

This thesis focuses mainly on Sentiment Analysis of patient reviews. While most existing sentiment analysis systems are predicated by POS (parts of speech) tagging or Bayesian sentiment analysis methods, the same cannot be applied to medical reviews as they generally carry a negative flavor in them. We thereby approached the problem by identifying the features in the sentence and calibrating the sentiment on a Negativity Meter based on their relation to sentiment words. A
feature, as defined for the purpose of this thesis, can be a medicine, a side effect or a symptom. The sentiment of each feature is determined by the aggregate of all its polarities with respect to each sentiment word, where the polarity is determined by an inverse relation to the distance of the feature from the sentiment word. Each sentence is then evaluated by the cumulative polarity of all the features contained in it. Sentiment of a review is determined by individually determining the sentiment of each sentence and then getting a weighted sum score of all the sentences in the review.

The accuracy of a sentiment analysis system is, in principle, how well it agrees with human judgments. Experimental results, involving human reviewers (extracted from site: www.askapatient.com) and correlating them back to the negativity rating of each review yield conclusive results, demonstrating the effectiveness of the technique. We have also implemented a customized Lucene search on the data using a multi-review summarization approach and a ranking scheme based on the feature-list. Ranking priority is given to the review that has the largest feature list size.
I would like to thank Prof. Tomasz Imielinski for all the support he has given me throughout the entire duration of my work under him. Right from the time of proposal of the idea, he gave me the flexibility to work towards the idea of my choice and simultaneously ensured that I was always on the right path. I have learnt invaluable lessons during the time I have worked under him on this thesis.

I would like to thank the my friends at Rutgers University for all their support and guidance during the course of my research. In particular I would like to thank Afaque Amanulla and Sangeetha Rajagopalan, Sangeetha for having worked with me to help develop the underlying system leading upto this thesis and it would not have been possible without her and Afaque for helping me with my statistical observations by participating in the human annotation process. Working with them has been a great learning experience for me.
This thesis is dedicated to my Grandparents. They have always been my friends and my gurus, who have guided me through the dark, into the correct path.
## Contents

Abstract  
Acknowledgements  
Dedication  
List of Figures  

1. Introduction

2. Sentiment Analysis
   2.1. Sentiment Analysis and the design choices  
   2.2. Discussion of the State-of-the-Art
      2.2.1. Approach 1: Graph based Models  
      2.2.2. Approach 2: Sentiment Scoring Systems  
      2.2.3. Approach 3: Classification methods
   2.3. Sideffectve: Sentiment Analysis

3. Search and Ranking
   3.1. ZEND LUCENE Indexing
      3.1.1. The Indexing Process  
      3.1.2. Analysis  
      3.1.3. Core Indexing classes
   3.2. Search  
   3.3. Ranking

4. Results
   4.1. Utilizing user reviews to calculate accuracy
   4.2. Independent review and annotation
      4.2.1. Kappa($\kappa$) Explained  
      4.2.2. Result of Evaluation

5. Conclusion
   5.1. Observations and Inferences
   5.2. Future Work
A Appendix: System Description for programmers

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1 Architecture</td>
<td>52</td>
</tr>
<tr>
<td>A.1.1 Phases of implementation</td>
<td>52</td>
</tr>
<tr>
<td>A.2 Architecture</td>
<td>54</td>
</tr>
<tr>
<td>A.3 Database Description</td>
<td>58</td>
</tr>
<tr>
<td>A.3.1 Table Design</td>
<td>58</td>
</tr>
<tr>
<td>A.4 Implementation Details</td>
<td>60</td>
</tr>
<tr>
<td>A.4.1 Words.pl</td>
<td>60</td>
</tr>
<tr>
<td>A.4.2 Links.php</td>
<td>62</td>
</tr>
<tr>
<td>A.4.3 New.php</td>
<td>64</td>
</tr>
<tr>
<td>A.4.4 Side.php</td>
<td>65</td>
</tr>
<tr>
<td>A.4.5 Forums.php</td>
<td>66</td>
</tr>
<tr>
<td>A.4.6 Thread.php</td>
<td>67</td>
</tr>
<tr>
<td>A.4.7 Loginform.php, Log.php and Logout.php</td>
<td>68</td>
</tr>
<tr>
<td>A.5 Sentiment Analysis, Search and Ranking</td>
<td>69</td>
</tr>
<tr>
<td>A.6 Conclusion</td>
<td>69</td>
</tr>
</tbody>
</table>
## List of Figures

1. Sentiment-Polarity classification data ........................................ 12
2. Shown Min-Cut Algorithm application ........................................ 15
3. News and Blogs Sentiment 1 ....................................................... 19
4. News and Blogs Sentiment 1 ....................................................... 19
5. Polarity and Classification of Political Data ................................. 22
6. Thesaurus Example for "helped" .................................................. 25
7. Example Review for Lexapro ..................................................... 26
8. Workflow of the Sentiment Analysis algorithm for Sideffective ........ 28
9. Zend Search Workflow ............................................................. 33
10. Code snippet showing how to create a LUCENE Index .................. 35
11. Zend Search Example ............................................................... 38
12. Zend Search Example 2 ............................................................. 39
13. Comparison of Ranking Mechanisms .......................................... 40
14. Examples of Human rating vs Sideffective’s Negativity Rating ......... 43
15. Fleiss’ Kappa ............................................................................ 45
1 Introduction

Two quite separate phenomena have defined the last 3 to 5 years in human history - The move to a more evidence based medicinal and clinical practice, and the rise of internet. People no longer go to the doctor or pharmacist first these days. They always go to the Internet. The Internet has become the prime source of information for people in this day and age. It is this idea that we wanted to understand and analyze, as this is one the most interesting questions - What does a patient/user think about a drug/medicine? We set out to build a system - Sideffective, which analyzes the reviews that users have put up all over the web and which can be used to understand the mindset of a medicine user. We set out to extract the side effects that the user experiences, the nature of the side effects for a group of drugs and so on. This thesis describes one aspect of the system, which is the sentiment analysis of the reviews.

Understanding the human mind has always been a major task of psychology. But programming a system to do that has also been one of the greatest challenges. When a person talks about pain in the abdomen and rash in the under arm, it is impossible for a layman to discern the implications of the medicine used. In this context various natural language processing techniques are available which analyze text and try to decipher what a person has suggested. But not up until now has there been a system which mines medicine reviews and identifies the sentiment associated with it and deliver acceptable accuracy. Medicine reviews are very important in todays world because the user of the medicine undergoes lots of pain, which he might want to convey to the world about that medicine. So a user who wants to understand the common sentiment of what people are talking about a medicine used, can just type the name of the drug and be bombarded with information about the medicine.

After optimizing the data model required for the system, the next step was to setup
the search framework. We had tested utilizing the Mysql InnoDB framework, but for reason explained in later sections, we had to drop the idea of adopting our own framework for search. So we utilized one of the most robust search frameworks, LUCENE. ZEND LUCENE offers a port to PHP, which was the server side technology utilized by us. Building the system involved many challenges that like identifying the various aspects to build the Side effect lexicon. But one challenge discussed in this thesis is the one of building the opinion lexicon. Opinion lexicons are generally built using WordNet and POS tagging of the data set. But we have discussed an innovative approach, which utilizes a thesaurus service and a training mechanism to achieve acceptable levels of completeness of the Opinion Lexicon.

To bring out the sentiment in a review, we need to identify what the person refers to in a sentence and also understand its relation to the search query that the user has given. We cannot think of a result set consisting diabetes medicines, when the user has given a search query, Fever and headache. The relevance of the search queries is as important as the relevance of the evaluated sentiment in the review. Therefore we have come up with a system that analyzes the opinion of each review by breaking it down to several levels and then evaluating based on certain factors, which we refer to as features. Features form an important part of Sideffective (Our sentiment analysis system). To evaluate the sentiment expressed in each review, we bring out the associations that the features have with the Opinion Lexicon and how they are weighted for each individual review. We also bring out the differences between our system and historically implemented Sentiment analysis systems.

Finally we have given our take on how to analyze a sentiment analysis system. When we build a system that parses human opinion, it is important to know how well it weighs against human rating of each review. This we have introduced in 2 ways, as we shall see discussed in the Results section of this thesis. With the innovative methodologies used, we try to bring out the accuracy and efficiency of
the system and its economical use of resources while at the same time delivering results that are of high standards. Statistical methods make use of Agreement Kappas ($\kappa$), to give solid conclusions about the agreement between human rating and calculated sentiment.

Though we have encountered many a challenge along the way, we have failed to resolve some issues, which we have talked about later on. These challenges posed are considerable mountains to climb and also do not allow us to attain higher accuracy levels. The System presents a good use case of how to mine highly opinionated data sets for useful and meaningful information and how to interpret the information. Application of such a system to political is another interesting area. Political data like medical data is highly opinionated, albeit having both sides of the coin. Medical data like this generally has a negative flavor, and hence the approach we have described makes complete sense in todays world.
2 Sentiment Analysis

2.1 Sentiment Analysis and the design choices

The Internet domain has morphed into a huge entity which contains data of extreme interest. Simple questions are interesting to look at. What does a person think of a movie? How did the product launch of the iPad go? Is the Nikon d3s an allround success? Did man ever get to the moon? There are many such questions on the internet. Todays internet domain has led to this interesting question of what is the Sentiment of a generic population regarding any topic of discussion, be it internet forums, websites, and most famous of all, Sentiment on a social networking website such as Twitter. Mining Twitter Sentiment is the order of the day and the results are astounding. We have set out to analyze a problem of a similar nature with our web miner. What is a persons opinion about a particular medicine?

The Web has aggressively changed the way that people express their views and opinions. Everyone can now post reviews of products at merchant sites and express their views on almost anything in online discussion groups, forums, and blogs, which are the user-generated content we are interested in. This expression of opinions represents new and measurable sources of information with many practical applications. Now if one wants to buy a product, he/she is no longer constrained to asking people physically. One has to only look up for such discussion over the internet to form opinions based on actual user experiences. For a company, it may no longer be necessary to conduct surveys, organize focus groups or employ external consultants in order to find consumer opinions about its products and those of its competitors because the user-generated content on the Web can already give them such information.

However, discovering opinion sources and analyze them over the Web can still be
a daunting task because of the sheer number of sources available, and also the fact that each source might have high volumes of opinionated text. In many cases, opinions are hidden in long forum posts and blogs. It is difficult for a human reader to find relevant sources, extract related sentences with opinions, read them, summarize them, and organize them into usable forms. Thus, automated opinion discovery and summarization systems are needed. Sentiment analysis, also known as opinion mining, grows out of this need. It is a challenging natural language processing or text mining problem.

The basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative or neutral. Early work in that area includes Turney(W) and Pang(W) who applied different methods for detecting the polarity of product reviews and movie reviews respectively. This work is at the document level. One can also classify a document’s polarity on a multi-way scale, which was attempted by Pang1(W) and Snyder(W) (among others), expanded the basic task of classifying a movie review as either positive or negative to predicting star ratings on either a 3 or a 4 star scale, and also performed an in-depth analysis of restaurant reviews, predicting ratings for various aspects of the given restaurant, such as the food and atmosphere (on a five-star scale).

Therefore a technique to detect favorable and unfavorable opinions toward specific subjects (such as organizations and their products) within large numbers of documents offers enormous opportunities for various applications. It would provide powerful functionality for competitive analysis, marketing analysis, and detection of unfavorable rumors for risk management. However, analysis of favorable and unfavorable opinions is a task requiring high intelligence and deep understanding of the textual context, drawing on common sense and domain knowledge as well
as linguistic knowledge. The interpretation of opinions can be debatable even for humans. For example, when we tried to determine if each specific review was on balance favorable or unfavorable toward a subject after reading an entire group of such reviews, we could achieve only a 74.4% rate of consensus, even for very small groups of evaluators. Therefore, we focused on finding local statements on sentiments rather than analyzing opinions on overall favorability.

Another direction is subjectivity/objectivity identification. This task is commonly defined as classifying a given text (usually a sentence) into one of two classes: objective or subjective. This problem can sometimes be more difficult than polarity classification, the subjectivity of words and phrases may depend on their context and an objective document may contain subjective sentences (e.g., a news article quoting people’s opinions). Moreover, as mentioned by Su(W), results are largely dependent on the definition of subjectivity used when annotating texts. However, Pang(W) showed that identifying objective sentences in a document before classifying its polarity helped improve performance.

The more fine-grained analysis model is called the feature/aspect-based sentiment analysis. It refers to the study of determining the opinions or sentiments expressed on different features or aspects of entities, e.g., a cell phone, a digital camera, insurance, or a bank. A feature or aspect is an attribute or a component of an entity, e.g., the screen of a cell phone, or the picture quality of a camera. This problem involves several sub-problems, e.g., identifying relevant entities, extracting their features/aspects, and determining whether an opinion expressed on each feature/aspect is positive, negative or neutral.

The latter model is what we use to determine the opinion expressed in the reviews that we focus on for the purpose of this thesis, Medicine reviews. Textual information in the world can be broadly categorized into two main types: facts and opinions. Facts are objective expressions about entities, events and their proper-
ties. Opinions are usually subjective expressions that describe peoples sentiments, appraisals or feelings toward entities, events and their properties. The concept of opinion is very broad. In this thesis, we only focus on opinion expressions that convey peoples positive or negative sentiments. Much of the existing research on textual information processing has been focused on mining and retrieval of factual information, e.g., information retrieval, Web search, text classification, text clustering and many other text mining and natural language processing tasks.

Sentiment analysis or opinion mining is the computational study of opinions, sentiments and emotions expressed in text. But the bottom line is, to decide what opinion to mine or to extract from a given text. Given a product review (which is the most general class of text evaluated in the opinion mining world), the sentences in the product can express either a positive, negative or neutral perspective. When we concentrate on the opinions expressed in each review, we need to understand the correlation of the opinion expressed, to an important factor in the review. This we call a feature. Building this relation would give us an idea about the reviews orientation with regards to the each feature and consequentially to all the features and finally to the sentiment orientation of the review itself. But feature-based sentiment analysis techniques also generally involve the sentiment orientation of the feature with respect to an object in context in the review. All other non-objective opinions are then considered to be of lesser weight.

Hence it is of utmost importance that we handle analysis for the purpose of this thesis subjectively. By discretely evaluating the sentences in the review and then subjectively classifying each sentence, identifying the opinion conveyed in each sentence of the review becomes a hugely simpler task and also conveys the exact opinion of each sentence. Evaluating the review subjectively is important in this respect because medicine reviews available over the Internet generally carry a negative flavor to them and therefore, cannot be evaluated as objectively. Objec-
tive evaluation would render 95% of the reviews negative. Also, when evaluated objectively, a review might say a medicine is effective. But the reviewer might have mentioned certain side effects that were a consequence of using the medicine. These side effects cannot be ignored and therefore need to be taken into account. Objective evaluation ignores such details. Evaluation of the medicines also as subjects of a sentence and not objects in a sentence helps the accuracy of evaluation of medicine.

By considering the medicines as features along with side effects and symptoms allows the system to evaluate the sentences subjectively and this helps in bringing out the true sentiment orientation of each sentence. Also this helps in evaluating topical relevance of the review with respect to a search query when used in conjunction with a search engine. Another advantage of using a feature list is its utility in ranking the reviews.

Creating systems that can process subjective information effectively requires overcoming a number of novel challenges. Let us now discuss what is involved in building a concrete sentiment analysis system. As we have discussed, such an application would fill an important and prevalent information need, whether one restricts attention to blog search or considers the more general types of search.

The development of a complete review- or opinion-search application might involve attacking each of the following problems:

1. If the application is integrated into a general-purpose search engine, then one would need to determine whether the user is in fact looking for subjective material. This may or may not be a difficult problem in and of itself: perhaps queries of this type will tend to contain indicator terms like review, reviews, or opinions, or perhaps the application would provide a checkbox to the user so that he or she could indicate directly that reviews are what is desired; but in general, query classification is a difficult problem.
2. Besides the still-open problem of determining which documents are topically relevant to an opinion-oriented query, an additional challenge we face in our new setting is simultaneously or subsequently determining which documents or portions of documents contain review-like or opinionated material. Sometimes this is relatively easy, as in texts fetched from review-aggregation sites in which review-oriented information is presented in relatively stereotyped format: examples include Epinions.com and Amazon.com. However, blogs also notoriously contain quite a bit of subjective content and thus are another obvious place to look (and are more relevant than shopping sites for queries that concern politics, people, or other non-products), but the desired material within blogs can vary quite widely in content, style, presentation, and even level of grammaticality.

3. Once one has target documents in hand, one is still faced with the problem of identifying the overall sentiment expressed by these documents and/or the specific opinions regarding particular features or aspects of the items or topics in question, as necessary. Again, while some sites make this kind of extraction easier for instance, user reviews posted to Yahoo! Movies must specify grades for pre-defined sets of characteristics of films more free-form text can be much harder for computers to analyze, and indeed can pose additional challenges; for example, if quotations are included in a newspaper article, care must be taken to attribute the views expressed in each quotation to the correct entity.

4. Finally, the system needs to present the sentiment information it has garnered in some reasonable summary fashion. This can involve some or all of the following actions:

a Aggregation of votes that may be registered on different scales (e.g., one reviewer uses a star system, but another uses letter grades).

b Selective highlighting of some opinions.
c Representation of points of disagreement and points of consensus.

d Identification of communities of opinion holders.

e Accounting for different levels of authority among opinion holders.

Note that it might be more appropriate to produce a visualization of sentiment data rather than a textual summary of it, whereas textual summaries are what is usually created in standard topic-based multi-document summarization.

The increasing interest in opinion mining and sentiment analysis is partly due to its potential applications, which we have just discussed. Equally important are the new intellectual challenges that the field presents to the research community. So what makes the treatment of evaluative text different from classic text mining and fact-based analysis?

Take text categorization, for example. Traditionally, text categorization seeks to classify documents by topic. There can be many possible categories, the definitions of which might be user- and application-dependent; and for a given task, we might be dealing with as few as two classes (binary classification) or as many as thousands of classes (e.g., classifying documents with respect to a complex taxonomy). In contrast, with sentiment classification, we often have relatively few classes (e.g., negative or 3 stars) that generalize across many domains and users. In addition, while the different classes in topic-based categorization can be completely unrelated, the sentiment labels that are widely considered in previous work typically represent opposing (if the task is binary classification) or ordinal/numerical categories (if classification is according to a multi-point scale). In fact, the regression-like nature of strength of feeling, degree of positivity, and so on seems rather unique to sentiment categorization (although one could argue that the same phenomenon exists with respect to topic-based relevance).

There are also many characteristics of answers to opinion-oriented questions that
differ from those for fact-based questions [284]. As a result, opinion-oriented information extraction, as a way to approach opinion-oriented question answering, naturally differs from traditional information extraction (IE) [49]. Interestingly, in a manner that is similar to the situation for the classes in sentiment-based classification, the templates for opinion-oriented IE also often generalize well across different domains, since we are interested in roughly the same set of fields for each opinion expression (e.g., holder, type, strength) regardless of the topic. In contrast, traditional IE templates can differ greatly from one domain to another—the typical template for recording information relevant to a natural disaster is very different from a typical template for storing bibliographic information.

These distinctions might make our problems appear deceptively simpler than their counterparts in fact-based analysis, but this is far from the truth. When we sample a few examples, we will be able to observe what makes these problems difficult compared to traditional fact-based text analysis.

Let us begin with a sentiment polarity text-classification example. Suppose we wish to classify an opinionated text as either positive or negative, according to the overall sentiment expressed by the author within it. Is this a difficult task?

The results of an early study by Pang et al. [W] on movie reviews suggest that coming up with the right set of keywords might be less trivial than one might initially think. The purpose of Pang et al.’s pilot study was to better understand the difficulty of the document-level sentiment-polarity classification problem. Two human subjects were asked to pick keywords that they would consider to be good indicators of positive and negative sentiment. As shown in Figure 1.

The use of the subjects lists of keywords achieves about 60% accuracy when employed within a straightforward classification policy. In contrast, word lists of the same size but chosen based on examination of the corpus statistics achieves almost 70% accuracy even though some of the terms, such as still, might not look that
intuitive at first.

However, the fact that it may be non-trivial for humans to come up with the best set of keywords does not in itself imply that the problem is harder than topic-based categorization. While the feature still might not be likely for any human to propose from introspection, given training data, its correlation with the positive class can be discovered via a data-driven approach, and its utility (at least in the movie review domain) does make sense in retrospect. Indeed, applying machine-learning techniques based on unigram models can achieve over 80% in accuracy [W], which is much better than the performance based on handpicked keywords reported above. However, this level of accuracy is not quite on par with the performance one would expect in typical topic-based binary classification.

In fact, the example that opens this section, which was taken from the following quote from Mark Twain, is also followed by a sentence with no ostensibly negative words:

You say I must familiarize my mind with the fact that Miss Austen is not a poetess, has no sentiment (you scornfully enclose the word in inverted commas), has no eloquence, none of the ravishing enthusiasm of poetry; and then you add, I must learn to acknowledge her as one of the greatest artists, of the greatest painters of
human character, and one of the writers with the nicest sense of means to an end that ever lived.

Note the fine line between facts and opinions: while Miss Austen is not a poetess can be considered to be a fact, none of the ravishing enthusiasm of poetry should probably be considered as an opinion, even though the two phrases s (arguably) convey similar information. Thus, not only can we not easily identify simple keywords for subjectivity, but we also find that like the fact that do not necessarily guarantee the objective truth of what follows them and bigrams like no sentiment apparently do not guarantee the absence of opinions, either. We can also get a glimpse of how opinion-oriented information extraction can be difficult. For instance, it is non-trivial to recognize opinion holders. In the example quoted above, the opinion is not that of the author, but the opinion of You, which refers to George Lewes in this particular letter. Also, observe that given the context (you scornfully enclose the word in inverted commas, together with the reported endorsement of Austen as a great artist), it is clear that has no sentiment is not meant to be a show-stopping criticism of Austen.

In general, sentiment and subjectivity are quite context-sensitive, and, at a coarser granularity, quite domain dependent (in spite of the fact that the general notion of positive and negative opinions is fairly consistent across different domains). Note that although domain dependency is in part a consequence of changes in vocabulary, even the exact same expression can indicate different sentiment in different domains. For example, go read the book most likely indicates positive sentiment for book reviews, but negative sentiment for movie reviews.

In the following sections, we aim to give an overview of a selection of some state-of-the-art efforts to address some of these issues, and march through the design choices made and a comparison against the design choices made for the purpose of this thesis.
2.2 Discussion of the State-of-the-Art

In this section we evaluate 7 sentiment analysis techniques that have been applied to varied domains in a bid to understand their approaches and to decipher their feasibility when applied to the medicine review domain. The techniques we evaluated were applied on data sets that contained product reviews/movie reviews/political opinions. When we consider the corpora mentioned, all of them are generally opinionated and we should note that reviewers could annotate them positively or negatively. Also, the lexicon generation techniques that we discuss here partly pave the way to build up the opinion lexicon for the system Sideffective.

2.2.1 Approach 1: Graph based Models

Prior work on sentiment analysis involves that of Pang and Lee [W], which applies a graph based approach to evaluate the review objectively. One can consider document-level polarity classification to be just a special (more difficult) case of text categorization with sentiment- rather than topic-based categories. Hence, standard machine learning classification techniques, such as support vector machines (SVMs), can be applied to the entire documents themselves, as was done by Pang, Lee, and Vaithyanathan [W]. Such classification techniques are referred to as default polarity classifiers. However, as noted above, they implied that it was possible to improve polarity classification by analyzing only objective sentences (such as plot summaries in a movie review). Their proposal was to first employ a subjectivity detector that determines whether each sentence is subjective or not: discarding the subjective ones creates an extract that should better represent a reviews objective content to a default polarity classifier.

Therefore their approach was to supply the algorithm with pair-wise interaction information, e.g., to specify that two particular sentences should ideally receive the same subjectivity label but not state which label this should be. Incorporating
such information is somewhat unnatural for classifiers whose input consists simply of individual feature vectors, such as Naive Bayes or SVMs, precisely because such classifiers label each test item in isolation. One could define synthetic features or feature vectors to attempt to overcome this obstacle. Hence the proposal of an alternative that avoids the need for such feature engineering: use of an efficient and intuitive graph-based formulation relying on finding minimum cuts. The main concern was physical proximity between the items to be classified; indeed, in computer vision, modeling proximity information via graph cuts has always led to very effective classification.

Subjectivity Classifiers were then used to apply the min cut once the correlation between the sentences were labeled. This approach has a completely objective view, which is not useful for the purpose of this thesis. The reason being, corpora of medical reviews are highly opinionated but contain a negative flavor. Hence it yields absolutely inaccurate results when applied to the dataset on hand. Especially since the approach uses a Nave Bayesian classifier as a subjectivity classifier and also for document-level polarity classification. This technique also focuses on other inputs such as context specific information with regards to the object and how the object attributes behave in the context of the review. This overhead is too large for a real time system to ignore even though we utilize caching mechanisms.
for the purpose of our miner. The subjectivity and polarity relations studied here paved the way for implementing the system we have on hand.

2.2.2 Approach 2: Sentiment Scoring Systems

One of the better approaches was proposed by Godbole, Srinivasaiah and Skiena et. al. In their system, which analyzes large scale news and blogs, they propose a sentiment scoring system that has been applied to improve the accuracy over graph based models.

Newspapers and blogs express opinion of news entities (people, places, things) while reporting on recent events. They present a system that assigns scores indicating positive or negative opinion to each distinct entity in the text corpus. The system consists of a sentiment identification phase, which associates expressed opinions with each relevant entity, and a sentiment aggregation and scoring phase, which scores each entity relative to others in the same class. The significance of our scoring techniques over large corpus of news and blogs is evaluated.

News can be good or bad, but it is seldom neutral. Although full comprehension of natural language text remains well beyond the power of machines, the statistical analysis of relatively simple sentiment cues can provide a surprisingly meaningful sense of how the latest news impacts important entities.

Several aspects of this type of sentiment analysis system, include:

1. Algorithmic Construction of Sentiment Dictionaries Sentiment index relies critically on tracking the reference frequencies of adjectives with positive and negative connotations. A method is presented for expanding small candidate seed lists of positive and negative words into full sentiment lexicons using path-based analysis of synonym and antonym sets in WordNet. Sentiment-alternation hop counts are used to determine the polarity strength of the candidate terms and eliminate the ambiguous terms. We present the detailed
2. Sentiment Index Formulation There is considerable subtlety in constructing a statistical index, which meaningfully reflects the significance of sentiment term juxtaposition. The presented technique is of using juxtaposition of sentiment terms and entities and a frequency-weighted interpolation with world happiness levels to score entity sentiment.

3. Evaluation of Significance The provided statistical evidence of the validity of sentiment evaluation by correlating the index with several classes of real-world events, including (1) results of professional baseball and basketball games, (2) performance of stock-market indices, and (3) seasonal effects [W]. Positive correlations prove that our sentiment analyzer can accurately measure public sentiment similar to the one provided by Godbole [W]. We also present additional anecdotal evidence corroborating our analysis.

Wiebe [8] evaluates adjectives for polarity as well as gradation classification. A statistical model groups adjectives into clusters, corresponding to their tone/orientation. The use of such gradable adjectives is an important factor in determining subjectivity. Statistical models are used to predict the gradability of adjectives.

Kim and Hovy [9] evaluate the sentiment of an opinion holder (entity) using WordNet to generate lists of positive and negative words by expanding seed lists. They assume that synonyms (antonyms) of a word have the same (opposite) polarity. The percentage of a word’s synonyms belonging to lists of either polarity was used as a measure of its polarity strength, while those below a threshold were deemed neutral or ambiguous. Their best results were achieved when the topic neighborhood consisted of words between the topic up to the end of the sentence.

This type of sentiment analyzer ignores articles, which are detected as being a duplicate of another. This prevents news syndicate articles from having a larger impact on the sentiment than other articles. Use of co-occurrence of an entity and
a sentiment word in the same sentence to mean that the sentiment is associated with that entity is common. This is not always accurate, particularly in complex sentences.

Several steps are taken to aggregate entity references under different names. By employing techniques for pronoun resolution, they identify more entity/sentiment co-occurrences than occur in the original news text. Further, Lydias system for identifying co-reference sets [W] associates alternate references such as George W. Bush and George Bush under the single synonym set header George W. Bush. This consolidates sentiment pertaining to a single entity.

They use the raw sentiment scores to track two trends over time:

1. Polarity: Is the sentiment associated with the entity positive or negative?

2. Subjectivity: How much sentiment (of any polarity) does the entity garner?

Subjectivity indicates proportion of sentiment to frequency of occurrence, while polarity indicates percentage of positive sentiment references among total sentiment references.

Focus is first on polarity. They evaluate world polarity using sentiment data for all entities for the entire time period and then evaluate entity polarity using sentiment data for that day only. In general, pairs of indices are positively correlated but not very strongly. This is not good, as it shows each subindex measures different things. The General index is the union of all the indices and hence is not positively correlated with each individual index.

The subjectivity time series reflects the amount of sentiment an entity is associated with, regardless of whether the sentiment is positive or negative. Reading all news text over a period of time and counting sentiment in it gives a measure of the average subjectivity levels of the world. They evaluate world subjectivity using sentiment data for all entities for the entire time period. They evaluate entity
Figure 3: News and Blogs Sentiment 1

<table>
<thead>
<tr>
<th>Actor</th>
<th>Net sentiment</th>
<th></th>
<th>Actor</th>
<th>Net sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Felicity Huffman</td>
<td>1.337</td>
<td>0.774</td>
<td>Joe Paterno</td>
<td>1.527</td>
</tr>
<tr>
<td>Fernando Alonso</td>
<td>0.977</td>
<td>0.702</td>
<td>Phil Mickelson</td>
<td>1.104</td>
</tr>
<tr>
<td>Dan Rather</td>
<td>0.906</td>
<td>-0.040</td>
<td>Tom Brokaw</td>
<td>1.042</td>
</tr>
<tr>
<td>Warren Buffett</td>
<td>0.882</td>
<td>0.704</td>
<td>Zach Cohen</td>
<td>1.000</td>
</tr>
<tr>
<td>Joe Paterno</td>
<td>0.881</td>
<td>1.527</td>
<td>Ted Stevens</td>
<td>0.820</td>
</tr>
<tr>
<td>Ray Charles</td>
<td>0.843</td>
<td>0.138</td>
<td>Rafael Nadal</td>
<td>0.787</td>
</tr>
<tr>
<td>Bill Frist</td>
<td>0.810</td>
<td>0.307</td>
<td>Felicity Huffman</td>
<td>0.774</td>
</tr>
<tr>
<td>Ben Wallace</td>
<td>0.778</td>
<td>0.570</td>
<td>Warren Buffett</td>
<td>0.704</td>
</tr>
<tr>
<td>John Negroponte</td>
<td>0.775</td>
<td>0.059</td>
<td>Fernando Alonso</td>
<td>0.702</td>
</tr>
<tr>
<td>George Clooney</td>
<td>0.724</td>
<td>0.288</td>
<td>Chauncey Billups</td>
<td>0.685</td>
</tr>
<tr>
<td>Alicia Keys</td>
<td>0.724</td>
<td>0.147</td>
<td>Maria Sharapova</td>
<td>0.680</td>
</tr>
<tr>
<td>Roy Moore</td>
<td>0.720</td>
<td>0.349</td>
<td>Earl Woods</td>
<td>0.672</td>
</tr>
<tr>
<td>Jay Leno</td>
<td>0.710</td>
<td>0.107</td>
<td>Jason Kidd</td>
<td>0.609</td>
</tr>
<tr>
<td>Roger Federer</td>
<td>0.702</td>
<td>0.512</td>
<td>Tom Brady</td>
<td>0.603</td>
</tr>
<tr>
<td>John Roberts</td>
<td>0.698</td>
<td>-0.372</td>
<td>Ben Wallace</td>
<td>0.570</td>
</tr>
</tbody>
</table>

Figure 4: News and Blogs Sentiment 1

| Actor          | Net sentiment |   | Actor          | Net sentiment |   |
|----------------|---------------|---|---------------|---------------|
| Slobozan Milosevic | -1.674      | -0.964 | John Muhammad | -3.076       |
| John Ashcroft   | -1.294       | -0.266 | Sammy Soa     | -1.702       |
| Zacarias Moussaoui | -1.239    | -0.908 | George Ryan   | -1.611       |
| John Muhammad   | -0.979       | -3.076 | Lionel Tate   | -1.112       |
| Lionel Tate     | -0.962       | -1.112 | Esteban Loaiza | -1.108     |
| Charles Taylor  | -0.818       | -0.302 | Slobdan Milosevic | -0.964   |
| George Ryan     | -0.789       | -1.511 | Charles Schumer | -0.949    |
| Al Sharpton     | -0.782       | -0.043 | Scott Peterson | -0.937       |
| Peter Jennings  | -0.781       | -0.372 | Zacarias Moussaoui | -0.908   |
| Saddam Hussein  | -0.652       | -0.240 | William Jefferson | -0.720   |
| Jose Padilla    | -0.576       | -0.534 | King Gyanendra | -0.620       |
| Abdul Rahman    | -0.570       | -0.500 | Ricky Williams | -0.603       |
| Adolf Hitler    | -0.549       | -0.159 | Ernie Fletcher | -0.589       |
| Harriet Miers   | -0.511       | 0.113 | Edward Kennedy | -0.879       |
| King Gyanendra  | -0.502       | -0.626 | John Gotti    | -0.554       |
subjectivity using sentiment data for that day only.

2.2.3 Approach 3: Classification methods

With the rise of weblogs and the increasing tendency of online publications to turn to message-board style reader feedback venues, informal political discourse is becoming an important feature of the intellectual landscape of the Internet, creating a challenging and worthwhile area for experimentation in techniques for sentiment analysis. We describe preliminary statistical tests on a new dataset of political discussion group postings, which indicate that posts made in direct response to other posts in a thread have a strong tendency to represent an opposing political viewpoint to the original post. We conclude that traditional text classification methods will be inadequate to the task of sentiment analysis in this domain, and that progress is to be made by exploiting information about how posters interact with each other.

As in the commercial domain, there are many applications for recognizing politically-oriented sentiment in texts. These applications include, among others, analyzing political trends within the context of a given natural language domain as a means of augmenting opinion polling data; classifying individual texts and users in order to target advertising and communications such as notices, donation requests or petitions; and identifying political bias in texts, particularly in news texts or other purportedly unbiased texts.

Many of the challenges of the present task are analogous, but not always identical, to those faced by traditional sentiment analysis. It is well-known that people express their feelings and opinions in oblique ways. Word-based models succeed to a surprising extent but fall short in predictable ways when attempting to measure favorability toward entities. Pragmatic considerations, sarcasm, comparisons, rhetorical reversals (I was expecting to love it), and other rhetorical devices tend to
undermine much of the direct relationship between the words used and the opinion expressed. Any task that seeks to extract human opinions and feelings from texts will have to reckon with these challenges. However, unlike opinion as addressed in conventional sentiment analysis, which focuses on favorability measurements toward specific entities, political attitudes generally encompass a variety of favorability judgments toward many different entities and issues. These favorability judgments often interact in unexpected or counterintuitive ways. In the domain of American politics, for example, it is likely that knowing a person's attitude toward abortion will help to inform a guess about that person's attitude toward the death penalty.

The first practical question which must be addressed is what specific information we are after and how to couch the task in terms of machine learning. We assume that we will approach this task as a classification task. So what are the classes? There is an element of arbitrariness in any selection of classes we might make. Political sentiment, as suggested above, is not a simple binary classification. Although the traditional right/left distinction is an obvious possibility, it is not enough to describe the various shades of American political thought. Other taxonomies exist which take into consideration more information, such as attitudes toward the structure and influence of government, personal and economic freedom, rationality, and other factors. It may not be necessary to model such nuances in practice, however. The classification scheme generally followed will need to reflect real divisions in the texts if it is to be modelable, but it will also depend largely upon practical considerations of what information we have decided we wish to extract. A related issue in practice is that of the kind of information we have available as training data.

Skeina conducted tests using several classification schemes. Use of both the hand-modified self-descriptions as they stood, and a more general classification of
right, left, and other, which was composed of people who described themselves as centrist, libertarian or independent is documented. The hand-modification mentioned on the self-descriptions was usually straightforward, although in one instance a self-described Conservative Democrat was modified to conservative. If there had been enough conservative Democrats in the data to justify it, this classification probably should have been allowed to stand as a distinct self-described class, and generalized to the other class.

To test the effectiveness of standard text classification methods for predicting political affiliation, the users were divided into the two general classes right (Republican, conservative, and r-fringe) and left (Democrat, liberal, and l-fringe), setting aside the centrist, independent, green, and libertarian users. Use of the naive Bayes text classifier to predict the political affiliation of a user based on the users posts is common practice in such type of classification. There were 96 users in the left category and 89 in the right, so a baseline classifier, which assigned the category, LEFT to every user would yield 51.89% accuracy. The NB text classifier gave an accuracy of 60.37% with a standard deviation of 2.21, based on 10-fold cross validation, While this is a statistically significant improvement over the baseline, it is modest.
Since purely text-based methods are unlikely to solve the problem of predicting political affiliations by themselves, we also looked at using the social properties of the community of posters. Unlike web pages, posts rarely contain links to other websites. However, many posts refer to other posts by quoting part of the post and then offering a response or by addressing another poster directly by name. This rule yields 77.45% accuracy for those users who quoted at least one post or had at least one post quoted by another user. However, since this covers only 55.7% of the users, this rule has an overall accuracy of only 65.57%, still an improvement over the NB classifier.

2.3 Sideffective: Sentiment Analysis

Sideffective is a system that does not rely on historically applied sentiment analysis methods. There has been a recent shift from determining the Sentiment based on the objectivity of a sentence and the sentiment orientation of the object in context. This can yield moderately accurate results for product reviews as they generally are focused on evaluating whether a product is good (positive) or bad (negative). Such type of evaluation is not sufficient in the evaluation of medicine review. In a medicine review, there are multiple other factors to consider. Consider the following, for example:

*with out Cymbalta my arms are numb and dead, can’t move them hardly at all. As for my side effects, my depression has worsen, the joint aches are severe, fatigue, constipation, diarrhea, vision problems, ringing in my ears, static and what I call electric strikes going through my head, and I have had suicidal thoughts and now I am having memory problems.*

This review is about a drug named Cymbalta. If this review is evaluated objectively by identifying just the objective sentence, then the review becomes a positive judgment. But when we run through the rest of the review, the user complains of
so many other side effects that he has suffered because of the act of partaking Cymbalta. Objective analysis omits this as there is no explicit mention of the fact that these side effects are a consequence of partaking Cymbalta. This is purely a logical implication. Hence we decided to evaluate the entire review subjectively, considering each sentence to be a subjective sentence (even the object The Medicine). The Algorithm makes use of a smooth flow of events that break down each review and analyzes it to evaluate whether a review is positive or negative.

For Sideffective, we have tried to applied the lessons learned in afore mentioned Sentiment Analysis systems in the design of the present system. One very useful technique that we grasped was to develop the Sentiment Lexicon. Sentiment/Opinion Lexicons are a common way to maintain a reference to the different sentiment or opinion expressing words that are part of the corpus. Sentiment lexicon generation has been achieved primarily by porting systems to WordNet historically. WordNet requires that the words to be modeled on be tagged by a POS (Parts-Of-Speech) tag. This tag has to be generated in the text by another system. This requires the use of another resource, which we felt was unnecessary. Instead we used a thesaurus service provided by BigHugeLabs. This thesaurus service enabled us to put in a small training data set and recursively build on the data set.

Algorithm 1: Opinion Lexicon Generation

Opinion Lexicon has been generated in many unsupervised learning mechanisms and has achieved considerable accuracy in prior work. But we have chosen a supervised approach where we use a small training data set to build the opinion lexicon. The training data set consisted of 50 positive and 50 negative words. The approach has given fair amount of accuracy and at the same time utilized an English language repository. Since it is important to follow language conventions while annotating a review as positive or negative, we have used a thesaurus service
Figure 6: Thesaurus Example for "helped"

provided by BigHugeLabs, which identifies the possible part of speech of the word and generates synonyms and antonyms of the word.

We separate the synonyms and antonyms and identify the unique words. The words are then made to undergo the above-described process on a recursive basis. This enables us to define a lexicon and label the words as having a positive polarity or negative polarity. The reason for following such a process is:

1. Avoid POS tagging POS tagging for a review dataset of about 400,000 reviews would have utilized considerable amount of resources and time. Hence we decide to use the thesaurus, which determines the part of speech given a word. This enabled us to editorially pick the adjectives and adverbs and recursively apply the algorithm. Historically WordNet has been able to provide the data
Lexapro worked for my depression and anxiety. But the side effects are horrible nausea, dizziness, sweating, tremors

Figure 7: Example Review for Lexapro

for such POS tagging events. (This has been the general approach of most Sentiment Analysis systems)

2. Reduce processing overhead As mentioned above, resource utilization is the primary focus here, as we do not want to use POS tagging for a corpus of 400000 reviews. POS tagging each review and identifying what part of the sentence is a descriptive word surely increases accuracy of generating individual words but does not ensure accurate contextual sentiment orientation. Also, our sentiment analysis approach does not involve any kind of POS utilization

This ensures that effective Lexicon generation is achieved. The accuracy of this approach has been compared against WordNet by taking a small corpus of 100 reviews and generating the Lexicon based on the 100 reviews using the WordNet approach and also using the approach described above. The difference in accuracy is approximately 2% (WordNet being marginally higher), which validates the fact that efficiency cannot be offset to achieve a marginally higher accuracy. Hence we have used the approach wherein we utilize a balance between accuracy and efficiency.

Algorithm 2: Sentiment Analysis

Now before we delve into the algorithm, let us try to examine the work-flow of the algorithm with the help of a small example.

Here we utilize what we call a feature set. A feature set can consist of a side effect, a symptom or a medicine. For the example here, the feature list is:
1. Lexapro
2. Depression
3. Anxiety
4. Nausea
5. Dizziness
6. Sweating
7. Tremors

The sole purpose of the Sentiment Analysis here is to determine the amount of negativeness of the review. This we measure on a normalized scale, which we call the Negativity Meter. The reasons behind this are:

1. Negative Flavor The medicine reviews that we reconciled from the web have been observed to contain a negative flavor to them. This renders the algorithm to be adapted to the quality of the review

2. A measure of how negative a medicine is when used with respect to side effects is always an eye-opener for anyone trying to use the medicine. This kind of analysis provides the user with the information he/she will be primarily looking for (How will the drug effect me?)

The Negativity Meter is a great representation of the sentiment a user wants to put forward. When the example is evaluated objectively, the review would suggest that the medicine Lexapro works and does not take into account the side effects as there is no objective relationship existing. When evaluated with the help of a feature list, the only thing we need to consider is the sentiment orientation of each feature in each sentence. This will give us an idea of what the sentiment of the whole review is.

We follow a breakdown model in evaluating the reviews. We single out each review
Figure 8: Workflow of the Sentiment Analysis algorithm for Sideffective

from the query hits. Once we know the review to be considered, then we break it down into sentences. It has been a general observation that the longer the review, the more negative it is. Isolating each sentence gives an idea of what the negativity of the sentence is. The evaluative method is as follows

Let us walkthrough the algorithm using the example discussed above. The algorithm performs a total of 3 evaluations:

1. The feature level
2. The sentence level
3. The review level

Feature level analysis determines the sentiment orientation of each feature in a given sentence. This we define as the primary focal point for this thesis. Each features sentiment/opinion orientation is determined as a function of the opinion words in the sentence. An opinion word has its influence on the feature depending on its proximity in the sentence. The farther the opinion word is to the feature,
the less the opinion it renders to the feature. This is the technique we use to evaluate every feature. The opinion words are identified upon their existence in the Opinion Lexicon discussed in the previous subsection.

We have assigned a constant polarity to all the opinion words identified by the Opinion Lexicon. If the opinion word is positive, the polarity is +1 and if it is negative, the polarity is -1. This helps us maintain a normalized evaluation of the sentiment orientation of each feature. The feature level analysis is as follows

\[ SO_f(f_i, s_j) = \sum_{w_k} \frac{SO(w_k)}{d(w_k, f_i)} \]

1. \( f_i \) is the feature in contexts, \( j \) is the sentence in which the feature is present

2. \( w_k \) is the opinion word used in correlation to the feature

3. \( d(w_k, f_i) \) is the distance of the opinion word from the feature

Here we can see that we isolate every feature in the sentence. Then, we build the opinion orientation for the feature considering all the opinion words in the sentence and their proximity to the feature. This ensures that we get the opinion orientation of each and every feature in the sentence.

The method uses sentiment lexicons to mark up all sentiment words and associated entities in the corpus. We reverse the polarity of a sentiment word whenever it is preceded by a negation.

\[ SO_s(s_i) = \sum_{f_j} SO(f_j, s_i) \]

1. \( s_i \) is the sentence in context

2. \( f_j \) is a feature present in the sentence

3. Summation over the sentiment orientation of all the features in the sentence
gives the sentiment orientation of the sentence

Once we have this data, we can build the opinion orientation of each sentence or what we call *Sentence Level* analysis. By computing the opinion orientation of each, it is possible to determine the overall opinion orientation each review. This is done in a weighted cumulative manner to adjust for inconsistencies in the size of the feature list per sentence. This we do because we do not want to leave an unfair advantage to a sentence, which is deemed to be 100% negative but has no feature in it while another sentence has a big feature list size but is only 50% negative. To offset this inconsistency, we have employed a weight in the cumulative evaluation, which makes the *Review Level* analysis.

\[
SO_r(r_i) = \sum_{s_j} SO_s(s_j) \cdot W_{s_j}(f)
\]

1. \(r_i\) is the review in context
2. \(s_j\) is a sentence present in the review
3. Here we use the weighted sum of the sentiment orientation of each sentence in the review
4. \(W_{s_j}(f)\) is the weight function of sentence \(s_j\), which is the size of the feature list for sentence \(s_j\)

The negativity of a review is represented on the *Negativity Meter* as mentioned previously. The Negativity Meter gives an accurate representation of the users sentiment expressed in the review. The accuracy analysis is discussed in the Results section. The more the number of blocks on the negativity meter, the more negative the review is.

The Sentiment Analysis approach we have adopted is completely catered to highly opinionated data sets such as medicine reviews. Such data sets have the capability to change the outlook the person reading them. So it is of utmost importance
that we handle these data sets carefully and provide accurate results as part of the analysis. If the accuracy is off by even 10% it gives an inaccurate representation about the medicine to the user of Sideffective. This has driven us to develop the system to adhere to latest statistical measures and thereby ensuring good quality in the results.
3 Search and Ranking

Lucene is an open source, highly scalable text search-engine library available from the Apache Software Foundation. Lucene’s powerful APIs focus mainly on text indexing and searching. It can be used to build search capabilities for applications such as e-mail clients, mailing lists, Web searches, database search, etc. Web sites like Wikipedia, TheServerSide, jGuru, and LinkedIn have been powered by Lucene.

Since most of the development of the system was web based, we opted to go the PHP route rather than the java route because it involves lesser setup time on the server side. This helped us implement the Java LUCENE port for the PHP framework called ZEND. ZEND LUCENE is a lucene search and indexing engine that has its libraries transferred over to the PHP space. So let us dive right into the development of the Search and Ranking algorithms using the customizable ZEND LUCENE engine.

These days, each web application requires FULLTEXT search. MySQL has a nice native implementation, PostgreSQL has one too, Lucene, Ferret (ruby port of Lucene) are just to name a few. However, while working on Sideffective, we faced a difficulty that MySQL’s InnoDB engine has: it doesn’t have FULLTEXT support. There is also no release date for this feature, giving us no choice but to look for an alternative. Zend provided us with the perfect search and indexing capabilities. The Lucene indexer is a very powerful tool which we used to develop a part of the precursor to this thesis, Sideffective: Side Effect Extraction That afore mentioned formed the basis for this thesis. The data modeling and data extraction was done as part of it and we had overcome many a challenge with regards to classification. But the indexer helped us classify the reviews and index them in the required format, which is the inverse term-document matrix, at the same time providing the required terms and document reference needed to build the lexicon of medical terminology. Medical terminology is necessary to define what is a side effect,
symptom or medicine and what is not one. For more information on the data modeling achieved for the system (Sideffective), please refer to Sideffective: Side Effect Extraction.

3.1 ZEND LUCENE Indexing

Lucene lets you index any data available in textual format. Lucene can be used with almost any data source as long as textual information can be extracted from it. You can use Lucene to index and search data stored in HTML documents, Microsoft Word documents, PDF files, and more. The first step in indexing data is to make it available in simple text format. You can do this using custom parsers and data converters.
3.1.1 The Indexing Process

*Indexing* is a process of converting text data into a format that facilitates rapid searching. A simple analogy is an index you would find at the end of a book: That index points you to the location of topics that appear in the book.

Lucene stores the input data in a data structure called an inverted index, which is stored on the file system or memory as a set of index files. Most Web search engines use an inverted index. It lets users perform fast keyword look-ups and finds the documents that match a given query. Before the text data is added to the index, it is processed by an analyzer (using an analysis process).

3.1.2 Analysis

*Analysis* is converting the text data into a fundamental unit of searching, which is called as term. During analysis, the text data goes through multiple operations: extracting the words, removing common words, ignoring punctuation, reducing words to root form, changing words to lowercase, etc. Analysis happens just before indexing and query parsing. Analysis converts text data into tokens, and these tokens are added as terms in the Lucene index.

Lucene comes with various built-in analyzers, such as SimpleAnalyzer, StandardAnalyzer, StopAnalyzer, SnowballAnalyzer, and more. These differ in the way they tokenize the text and apply filters. We have used what is called a Shingle Analyzer. Please refer to *Sideeffective: Side Effect Extraction* for more information on the analyzer used. As analysis removes words before indexing, it decreases index size, but it can have a negative effect on precision query processing. You can have more control over the analysis process by creating custom analyzers using basic building blocks provided by Lucene.

Index creation and updating capabilities are implemented within the Zend_Search_Lucene
3.1.3 Core Indexing classes

1. Directory - An abstract class that represents the location where index files are stored. There are primarily two subclasses commonly used:
   a. FSDirectory - An implementation of Directory that stores indexes in the actual file system. This is useful for large indices
   b. RAMDirectory - An implementation that stores all the indices in the mem-

Figure 10: Code snippet showing how to create a LUCENE Index component, as well as the Java Lucene project. You can use either of these options to create indexes that Zend_Search_Lucene can search.
ory. This is suitable for smaller indices that can be fully loaded in memory and destroyed when the application terminates. As the index is held in memory, it is comparatively faster

2. Analyzer - As discussed, the analyzers are responsible for preprocessing the text data and converting it into tokens stored in the index. IndexWriter accepts an analyzer used to tokenize data before it is indexed. To index text properly, you should use an analyzer that’s appropriate for the language of the text that needs to be indexed. Default analyzers work well for the English language. There are several other analyzers in the Lucene sandbox, including those for Chinese, Japanese, and Korean

3. IndexDeletionPolicy - An interface used to implement a policy to customize deletion of stale commits from the index directory. The default deletion policy is KeepOnlyLastCommitDeletionPolicy, which keeps only the most recent commits and immediately removes all prior commits after a new commit is done

4. IndexWriter - A class that either creates or maintains an index. Its constructor accepts a Boolean that determines whether a new index is created or whether an existing index is opened. It provides methods to add, delete, or update documents in the index

The changes made to the index are initially buffered in the memory and periodically flushed to the index Zend_Directory. IndexWriter exposes several fields that control how indices are buffered in the memory and written to disk. Changes made to the index are not visible to IndexReader unless the commit or close method of IndexWriter are Zend_Called. IndexWriter creates a lock file for the directory to prevent index corruption by simultaneous index Zend_Updates. IndexWriter lets users specify an optional index deletion policy.

Once the Indices are created, we can apply the search functionality to retrieve
the documents based on the search key and relevance from the inverted index association.

### 3.2 Search

Searching is the process of looking for words in the index and finding the documents that contain those words. Building search capabilities using Lucene’s search API is a straightforward and easy process. The following are the primary classes from the Lucene search API.

1. **Searcher** - Searcher is an abstract base class that has various overloaded search methods. IndexSearcher is a commonly used subclass that allows searching indices stored in a given directory. The Search method returns an ordered collection of documents ranked by computed scores. Lucene calculates a score for each of the documents that match a given query. IndexSearcher is thread-safe; a single instance can be used by multiple threads concurrently.

2. **Term** - Term is the most fundamental unit for searching. It’s composed of two elements: the text of the word and the name of the field in which the text occurs. Term objects are also involved in indexing, but they are created by Lucene internals.

3. **Query and subclasses** - Query is an abstract base class for queries. Searching for a specified word or phrase involves wrapping them in a term, adding the terms to a query object, and passing this query object to IndexSearcher’s search method. Lucene comes with various types of concrete query implementations, such as TermQuery, BooleanQuery, PhraseQuery, PrefixQuery, RangeQuery, MultiTermQuery, FilteredQuery, SpanQuery, etc. The section below discusses primary query classes from Lucene’s query API.

4. **TermQuery** - The most basic query type for searching an index. TermQuery
can be constructed using a single term. The term value should be case-sensitive, but this is not entirely true. It is important to note that the terms passed for searching should be consistent with the terms produced by the analysis of documents, because analyzers perform many operations on the original text before building an index.

5. QueryParser - QueryParser is useful for parsing human-entered query strings. You can use it to parse user-entered query expressions into a Lucene query object, which can be passed to IndexSearcher’s search method. It can parse rich query expressions. QueryParser internally converts a human-entered query string into one of the concrete query subclasses. You need to escape special characters such as *, ? with a backslash (\). You can construct Boolean queries textually using the operators AND, OR, and NOT.

For example, consider the e-mail subject "Job openings for Java Professionals at Bangalore." Assume you indexed this using the StandardAnalyzer. Now if we search for "Java" using Zend_Search_Lucene_Search_QueryParser, it would not return anything as this text would have been normalized and put in lowercase.
Figure 12: Zend Search Example 2

by the StandardAnalyzer. If we search for the lowercase word ”java,” it would return all the mail that contains this word in the subject field. Hence the use of ShingleAnalyzer, which takes care of this issue.

The search result is an array of Zend_Search_Lucene_Search_QueryHit objects. Each of these has two properties: $hit->id is a document number within the index and $hit->score is a score of the hit in a search result. The results are ordered by score (descending from highest score). The Zend_Search_Lucene_Search_QueryHit object also exposes each field of the Zend_Search_Lucene_Document found in the search as a property of the hit. In the following example, a hit is returned with two fields from the corresponding document: title and author. Optionally, the original Zend_Search_Lucene_Document object can be returned from the Zend_Search_Lucene_Search_QueryHit. You can retrieve stored parts of the document by using the getDocument() method of the index object and then get them by getFieldValue() method.

```php
1. $index = Zend_Search_Lucene::open('/data/my_index');
2. 
3. $hits = $index->find($query);
4. foreach ($hits as $hit) {
5.   // return Zend_Search_Lucene_Document object for this hit
6.   echo $document = $hit->getDocument();
7. 
8.   // return a Zend_Search_Lucene_Field object
9.   // from the Zend_Search_Lucene_Document
10. echo $document->getField('title');
11. 
12.   // return the string value of the Zend_Search_Lucene_Field object
13. echo $document->getFieldValuer('title');
14. 
15.   // same as getFieldValue()
16. echo $document->title;
17. }
```
3.3 Ranking

Ranking for the purpose of the thesis has been customized to include a very simple scheme to go in conjuncture with the ZEND LUCENE Search. If we recall the usage of a feature list in the sentiment analysis of a review, the same list can be used to determine the ranking order of the search results. Once we understand what the search results are and we reconcile the feature list for the each review, we can directly rank the search hits in the decreasing order of feature list size. This ranking scheme is the simplest and fastest when applied across the dataset. Following is a confirmation of this statement.

The ranking scheme utilizes the feature list to determine what review has to be ranked in what order. General rules of ranking followed specific to the thesis are:

1. When Drugname is used in search query, all relevant search hits of drug are ranked with higher priority over reviews of other drugs.

<table>
<thead>
<tr>
<th>Feature-List Ranking</th>
<th>Search Seconds:</th>
<th>0.036</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocName Seconds:</td>
<td>0.039</td>
<td></td>
</tr>
<tr>
<td>Num Points:</td>
<td>44.000</td>
<td></td>
</tr>
<tr>
<td>Num Good Points:</td>
<td>5.000</td>
<td></td>
</tr>
<tr>
<td>Max Good Points:</td>
<td>5.000</td>
<td></td>
</tr>
<tr>
<td>Average Precision:</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>MRR:</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>Recall:</td>
<td>1.000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tf-Idf Ranking</th>
<th>Search Seconds:</th>
<th>0.039</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocName Seconds:</td>
<td>0.045</td>
<td></td>
</tr>
<tr>
<td>Num Points:</td>
<td>44.000</td>
<td></td>
</tr>
<tr>
<td>Num Good Points:</td>
<td>5.000</td>
<td></td>
</tr>
<tr>
<td>Max Good Points:</td>
<td>5.000</td>
<td></td>
</tr>
<tr>
<td>Average Precision:</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>MRR:</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>Recall:</td>
<td>1.000</td>
<td></td>
</tr>
</tbody>
</table>
2. Proximity of search terms is taken into account with reviews having closer proximity of search terms having higher priority.

3. Ranking priority does not depend on the negativity of a review. This is because the negativity of the review should be a direct function of being ranked higher. When the negativity is a factor, it implies that the more number of side effects mentioned, the more the negativity is. But this is not always the case as the user might use a simple negation right next to the side effects mentioned.

We had made an attempt at designing our own search functionality using MYSQL INNODB as mentioned at the beginning. But due to lack of FULLTEXT search functionality and also lack of resources to optimize the search, we had to shift focus to LUCENE to help us in our search needs. Efficiency of the system is discussed in the next section.
4 Results

A Sentiment Analysis system is always evaluated against human review. We used two evaluative methods to determine the accuracy of the system. Both the methodologies use considerable human rating comparison and the agreement calculated between the ratings and the ratings provided by the Sentiment Analysis algorithm validate the accuracy of the algorithm. According to Pang[W], a sentiment analysis algorithm is deemed to be accurate if it agrees with human ratings 70% of the time. The evaluation techniques we have used are described in the following subsections.

4.1 Utilizing user reviews to calculate accuracy

The primary method of evaluation was the usage of ratings given by users over the Internet. Websites like www.askapatient.com, www.webmd.com, etc. have provision for user rating of a medicine. This was helpful while evaluating the accuracy of the algorithm. The algorithm provides a rating that is inversely proportional to the rating provided by the users in the websites mentioned above. This is because we measure the negativity of a medicine while the website give provision for effectiveness of a medicine. Hence this led us to consider the Spearman Correlation while evaluating the accuracy. The evaluation was done by scrapping 400 user reviews about 12 different medicines from 3 websites. Once we had the user ratings of each, we compared the ratings provided against the Negativity Rating provided by Sideffective. This allowed us to calculate the agreement between the two ratings using a Spearman Correlation. Although it is easy to program a spearman correlation calculator, there are many available over the web and therefore we decided to avail of the services of one such calculator, http://www.wessa.net/rankcorr.wasp. By providing the necessary rank correlation or even by providing raw data as
The data set that we considered had reviews with at least 50 words. This number was chosen so that the review had enough features to contribute towards a fair judgment of the negativity rating. The result of the Spearman Correlation in this case was 91.3%. This relates to saying that the negativity rating agrees 91% of the time with human ratings. Following are examples of the ratings provided.

When evaluated against a data set of words that have at most 30 words, the Correlation shot up to 97%. Even though this method yielded comprehensive results, the evaluation methodology cannot be applied to the entire data set because not all reviews have user provided ratings. To counter this drawback, we have employed the method described in the next section.

<table>
<thead>
<tr>
<th>Review</th>
<th>Askapatient rating</th>
<th>Negativity rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>I've been on Cymbalta for months now, 120 mg per day, and the only side effect I had when I started was drowsiness. I still get tired now if I take it in the morning, but since I've gone to taking it at night, the drowsiness has gotten much better</td>
<td>3</td>
<td>2.5 stars</td>
</tr>
<tr>
<td>Ambien 4 yrs Ambien CR 1yr memory loss, I don't sleep very long. Sometimes I will close my eyes and I can't move or speak, headache. I don't know how long it last. I can hear what's going on around me, but I can't call out for help. depression can neurontin be taken with cymbalta</td>
<td>1</td>
<td>4.5 stars</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0 stars</td>
</tr>
</tbody>
</table>
4.2 Independent review and annotation

In this evaluative approach, we set up a review panel comprising of 3 people. The 3 people were tasked with providing what they think might be the severity of a review. They were provided with a data set of 100 reviews and were asked to provide the ratings. Once the ratings from the review committee were received, the next task was to identify the correlation between them. But here Spearman correlation cannot be applied because, the agreement among the review committee has to be over a 70% agreement range (J.Fleiss[W]). If the agreement between the ratings provided by the review committee is not 70%, then the data set is deemed to be useless. Then a new data set has to be constructed for review rating purposes. Only after knowing this can we calculate the agreement among the ratings and negativity rating. For this purpose, we employed what is called a Kappa, Fleiss's Kappa to be precise.

4.2.1 Kappa(κ) Explained

Fleiss’ kappa (named after Joseph L. Fleiss, κ) is a statistical measure for assessing the reliability of agreement between a fixed number of raters when assigning categorical ratings to a number of items or classifying items. This contrasts with other kappas such as Cohen’s kappa, which only work when assessing the agreement between two raters. The measure calculates the degree of agreement in classification over that which would be expected by chance and is scored as a number between 0 and 1.

Fleiss’ kappa(κ) is a generalisation of Scott’s pi statistic, a statistical measure of inter-rater reliability. It is also related to Cohen’s kappa statistic. Whereas Scott’s pi and Cohen’s kappa work for only two raters, Fleiss’ kappa works for any number of raters giving categorical ratings (see nominal data), to a fixed number of items. It can be interpreted as expressing the extent to which the observed amount of
agreement among raters exceeds what would be expected if all raters made their ratings completely randomly. It is important to note that whereas Cohen’s kappa assumes the same two raters have rated a set of items, Fleiss’ kappa specifically assumes that although there are a fixed number of raters (e.g., three), different items are rated by different individuals (Fleiss, 1971, p.378). That is, Item 1 is rated by Raters A, B, and C; but Item 2 could be rated by Raters D, E, and F.

Agreement can be thought of as follows, if a fixed number of people assign numerical ratings to a number of items then the kappa will give a measure for how consistent the ratings are. The kappa, $\kappa$, can be defined as,

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$$

The factor $1 - \bar{P}_e$ gives the degree of agreement that is attainable above chance, and the factor $\bar{P} - \bar{P}_e$ gives the degree of agreement actually achieved above chance. If the raters are in complete agreement then $\kappa = 1$. If there is no agreement among the raters (other than what would be expected by chance) then $\kappa < 0$.

Kappa calculators are freely available over the Internet. We have used a matlab library that is available for free download at MATLAB CENTRAL in the MathWorks website.

4.2.2 Result of Evaluation

The evaluation method we have undertaken involved a 3 independent review committee as mentioned earlier. Once the committee had reviewed and assigned ratings to the 100 reviews, we then calculated the Fleiss Kappa of agreement. The data set was utilized because the Kappa of agreement came out to be 0.7435 or 74.4% agreement between the reviewers. The agreement Kappa for the Negativity Rating
and the Average of the user provided rating was calculated. This time the agree-
ment Kappa used was Cohens Kappa which is used to compute agreement when
there are only 2 factors. The Kappa in case turned out to be 0.89376 or 89.4%
agreement. This kappa of agreement is approximately equal to the 91.3% accuracy
achieved in the previous technique.

The 2 methods of evaluation afore mentioned are very robust as Statistical measure
is always the prime evaluative method when it comes to information retrieval and
processing. Hence this validates the fact that we have a very accurate Sentiment
Analysis system.
5 Conclusion

5.1 Observations and Inferences

Though there are various directions to be pursued when it comes to Sentiment Analysis, we are more interested in pursuing how the sentiment weighs against real world judgment of the Sentiment of a user review. Repositories of such data have been discussed in the work, *Sideeffective: Side Effect Extraction*. The direction that we employ is subjective in nature and so eliminates the possibilities of having situational sentiment analysis similar to product reviews. This type of analysis can only be applied to corpuses which have are highly opinionated. Such types of corpuses generally have very less or no middle ground at all. So one application of this work could be to Political Data.

Also, Sideffective gives us insights into how to analyze data that has been human annotated and compare it against the data that has been annotated by the algorithm with the negativity rating. Comparison of these ratings has been done in a statistically significant manner and we have also shown that we have observed and produced the accuracy of the method in its true form. The accuracy of the sentiment analysis methodology is the most important aspect here and we have shown that the accuracy is upto approximately 91% accurate.

The ideology of analyzing the sentiment surrounding such medicines is a big area of research presently. This is because more and more pharmaceutical companies are interested in finding out what users think about their product. We would therefore like to conclude by saying that these findings can pave way to more accurate representations of the Sentiment Analysis of highly opinionated medicinal reviews and the data surrounding it.
5.2 Future Work

By way of observing the results so described above, it is apparent that more statistical analysis needs to be done on this area of research. There were 2 primary issues that were not dealt with. They are:

1. Observing rare side effects that could potentially have influence on the negativity

2. Opinion Spam

The factor provided by rare side effects is one thing that can influence the negativity rating represented as part of this thesis. Rare side effects can be of grave nature or can be of a useless quality. Side effects like death, paralysis, etc. can be factors that can influence the negativity rating drastically as such a side effect is extremely grave in nature. So mining for such side effects and rules association for such mining was never made. This is viewed as a potential enhancement for this kind of research.

Opinion Spam is a major annoyance that can weigh against the negativity rating being accurate. Our judgment is that if Opinion Spam is eliminated from the review corpus, then the Negativity Rating can be enhanced in accuracy by at least a factor of 3%. Opinion Spam can be of the following type:

\[ \text{crohns/abcesses depression, metallic taste, dark urine, absolutely no energy, confusion, the list goes on, but as Im still on it I havent the energy to think!!!!! I am still on it and look forward to the day I can be off it and be my happy self again} \]

If we observe the above review that the user has provided about Depo-Provera, the user complains of many side effects as a consequence of using it. But when you actually observe the rating the user has given for the medicine (scraped off www.askapatient.com), the user has given a rating of 4. This suggests the medicine works for its intended use, but the user complains of many side effects. This is what we consider as Opinion Spam. Most users in general mention the medicines
usage as negative if they observe such side effects even though the medicine was effective in its intended cause.
6 Bibliography

1. *Opinion Mining and Sentiment Analysis (Early Text)*, Bo Pang and Lillian Lee

2. *A Sentimental Education: Sentiment Analysis Using Objectivity Summarization Based on Minimum Cuts*, Bo Pang and Lillian Lee

3. *A Preliminary Investigation into Sentiment Analysis of Informal Political Discourse*, Tony Mullen and Robert Malouf

4. *Large-Scale Sentiment Analysis for News and Blogs*, Namrata Godbole, Manjunath Srinivasaiah and Steven Skiena

5. *Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis*, Theresa Wilson, Janyce Wiebe and Paul Hoffmann


8. *Sentiment Analysis of Figurative Language using a Word Sense Disambiguation Approach*, Vassiliki Rentoumi and George Giannakopoulos

9. *Opinion Spam and Analysis*, Nitin Jindal and Bing Liu

10. *Mining Comparative Sentences and Relations*, Nitin Jindal and Bing Liu

11. *Mining and Summarizing Customer Reviews*, Minqing Hu and Bing Liu

12. *Sentiment Analysis of Conditional Sentences*, Ramanathan Narayanan and Alok Choudhary


15. *Learning with Compositional Semantics as Structural Inference for Subsentential Sentiment Analysis Techniques*, Yejin Choi and Claire Cardie
A Appendix: System Description for programmers

Aggregation is the buzzword for today's Internet-based applications, and that is the strength of this project as well. The past few years have seen tremendous growth in the Web, giving rise to innumerable opportunities for different creative undertakings. The amount of data available about any field is unimaginable and can be used to develop quite innovative applications that can be widely appreciated by a very diverse audience as well. The present project is also an effort in the same direction. This is an incremental effort and therefore features are being constantly updated to add more value to the application.

A.1 Architecture

Sideffective is a project aimed at creating a web-based application for the users on the Internet to access, search, and interact about medicines and their effects. The effort is towards collecting as much data as possible from various medical websites and harvest the user reviews in a useful manner, which is both visually appealing as well as easy to use with inherently valuable information. This document presents the technical features and implementation details included so far in this project.

A.1.1 Phases of implementation

The project has shaped through a series of steps, each involving a different methodology, software, or programming platform. This section gives a brief overview of the chronology of phases involved in the project development.

- Crawling: A number of websites containing relevant data were chosen based on the criteria that the pages essentially contain testimonials/reviews from the users of different drugs. Each of these websites is examined thoroughly so that only the necessary pages can be crawled instead of extracting the entire website.
which is very time consuming. A pattern in the URL is established and this URL pattern is supplied as input to the Web Crawler. In this case, the crawler used was HTTrack. This crawler has proved to be quite useful, as most of the restrictions placed by the administrators of the websites can be overridden. The output from this crawling process is a folder with a number of relevant HTML pages. This is the raw data as such which needs to be further processed through quite a few stages.

- Parsing: The files obtained from the previous step are .html files. They obviously contain html tags and a lot of metadata information which is not necessary to the context in concern. Therefore, Java's HTMLParser package has been used to extract the content (without the html tags) into text files, which can be processed further to form Database tables.

- Database Tables: Construction of Database tables in the next important task. Once the text files are generated, the next task is to insert the reviews/testimonials along with all other information into a table. This table in our context is xyz. It contains attributes like (DrugName, User, Link, Review). The only issue here is, each website would be organized differently, thereby making it difficult to reuse the same code modules to perform the extraction-insertion process. Therefore, the Perl code for each website varies based on the structure of the pages. The next tables to be constructed are the Unigram table (tabledsf) and Bigram table (bitable) along with calculations for frequencies of occurrence of words in the reviews. This is done is the Perl file named words.pl, which has been further explained in the later sections. The other tables like userdb and thread are constructed dynamically as it contains user data. As and when users add new information to the Forums feature of the application, these tables are populated.

- Interface and Algorithm: The final step is to use all the data accumulated to
build an application model that is most suitable to the users of such websites. An appropriate and easy-to-use, self-explanatory user interface is also very important as it is the first peek into the application for all the users. The suitable algorithms for searching, querying, Sentiment Analysis and presentation of data are incorporated within the interface. The following sections present a deeper explanation of the features as well the implementation details.

A.2 Architecture

This section describes the different features of the interface without dwelling much into the technical and implementation details. The next section describes in detail about all the files and techniques of the interface. The purpose of this section is purely to walk through the entire website to gain familiarity to the subject in discussion.

The home page of the website begins at index.php. There are several sections in this page itself. The first section is the menu bar on the top, which carries links to Home, About Us and Forums.

• Home, as the name suggests is the link which redirects to the homepage index.php from anywhere in the website. This menu bar is common to all the sub-pages in this website.

• About Us: This section gives a brief introduction about the team that was involved in this project development.

• Forums: Although most of the data has been collected by crawling various websites on the Internet, there is also the option of users entering data in this website which can be used for all other analysis. Thus, this link redirects to the Forums page where users can start new discussion threads and also participate in existing ones.
The next feature is right below the menu bar, that is, the new/existing User Login. This link redirects to the login page where the user enters his/her credentials to sign in. It is also possible for a new user to register to the Forums in this page.

The next section is the Header bar, which has two features. One, obviously carries the name of the Project in the center. The second, is the Alphabet List which acts as an index to the Drug names. This is important as it makes it easy to navigate to any drug beginning with a particular letter of the alphabet. The Header bar is also common for all pages of the website.

Right below the header bar, is the all-important Search Bar. Search can be done based on any Drug name or any side effect. Results of the search are displayed in a different page whose features are explained in the next section. The search bar is in the center only in the Index page. In all the other pages, it has been moved to the top right corner in the menu bar. The functionality remains the same though.

Another small section below this, termed as the **Funky Facts** is a set of observations made manually by examining the data and the its various peculiarities. This section can be changed as and when newer observations come to focus.

Finally, there is the Footer bar at the bottom. This carries the Acknowledgements, Project team members and the Disclaimer note. The footer bar remains unchanged in all the pages of the website.

The pages of the form list.php (where = a,b,cz) has the following structure: The Menu bar, Search Bar, Header and Footer are the same as described in the previous section. The body of the page has all the drugs in the database beginning with that letter arranged in alphabetical order.

The next page under discussion is the page, which appears when any drug link is clicked from list.php or if any Drug name is typed in the search bar (say drug name is xxx). The implementation of these features is present in the files links.php
and case2.php (first half) respectively. Again, the usual set of Menu bar, Search bar, Header and Footer are the same. The main body has two sub-sections.

- The left sub section gives the total number of reviews for this particular xxx drug and also a graphical representation of certain facts and figures. The first graph is a pie chart that shows the percentage distribution of the top 20 side effects for the drug xxx. The second graph is a slightly different interpretation of the frequencies from the database tables. It is a Comparative Bar graph for the frequencies of the top 10 side effects of xxx with respect to the frequencies of the side effect for the entire Drug database.

- The right sub section has the list of Top 20 Side effects experienced by users taking the drug xxx. Each of these side effects (say sss) are links which when clicked, generates user reviews reported in the database for drug xxx and side effect sss. These reviews are displayed on the left sub section for ease of navigation between various side effects. The right sub section also has a red link at the bottom that says Show all Side Effects. Since only the top 20 effects are displayed, this link can show all the reported side effects on the left. Along with the effect, the table also shows a ratio that denotes the number of users who have reported this side effect for this particular drug Vs the number of users reporting this side effect for all the drugs.

Note that the previous section describes the features of the webpage when a Drug name is either selected or searched for. The other case is, when any other key (say a symptom/effect called kkk) is searched for in the Search bar. Then the file that is invoked is the other half of case2.php. The page has the usual features and has a left and right sub section just like the previous case. The difference here is, the right sub section now displays the Top 20 Drugs which report the effect kkk. They are displayed as clickable links that show the reviews corresponding to the effect kkk and drug name as clicked.
The purpose of having both is facilitate a two way association wherein not only can users search for side effects given a drug, but also form an idea about which drug may possibly be causing a certain effect. It is quite a practical feature desired by a lot of users who have trouble associating a certain effect to a particular drug.

This next part describes the Forums section of the interface. All the data, as of now has been accumulated from various medicine related websites on the internet, it is wise to expect that users would post reviews or questions related to a particular drug in the same website as well. To harvest such data, the Forums section has been included. Following are some of its features:

• forums.php is the first peek into this section. Along with the regular features of the page like Menu bar, Search bar etc., this page displays a table. The table has two columns: Name/Topic of the discussion thread and no. of User comments to that thread. Each of the topics is a clickable link that redirects to the contents of that particular thread. Another important feature is the Start New thread button on top, which allows users to begin a new discussion thread. This requires the user to be signed in. This button redirects to login page if no user is signed in for that session else takes to the threadaddform.php.

• threadaddform.php simply has a form which takes in the username, Thread name and Thread content added by the signed in user. This form, on submission adds the new thread to the list of threads and it can be seen and replied to, by other users.

• thread.php is the page which comes up when any of the discussion threads is clicked. It contains all the comments posted by the users along with the Username. It is possible to add a comment to the existing thread if the user is signed in.

• Finally, logout link comes up on every page after login has been successful to facilitate logout.
A.3 Database Description

All the data extracted from the web (in the form of user reviews and testimonials) along with other details like username, link to that review on the web, Drug for which the review is given etc., is maintained in the MySQL database called DRUG. This database has multiple tables for various purposes. Each of the tables and their description is given below. The next section discusses the construction of these tables.

A.3.1 Table Design

- xyz This is the most important table which has the following structure:

  - drugname varchar(50): As the name suggests, this holds the name of the drug for which the review corresponds.
  - user varchar(150): Most websites which were crawled for the data were forums wherein the users had a unique username. This field does not carry a special functionality and has been included just for the sake of ease of implementation.
  - link varchar(500): This field carries the link to the website from which the particular review has been extracted. A small detail to be noted here is that, the links in case of some reviews may not match now as the users and administrators are constantly updating the websites, thereby posing the problem of outdated links.
  - review varchar(2000): This field is the most vital attribute as it has the actual testimonial written by the user for the specific drug. Important detail to be noted here is, the text is highly unstructured with no specific format. Therefore, cleansing the data sufficiently is very important before using it for any analysis.

- bitable This table has the various drugs and correspondingly, the different
bigram side effects found in their user reviews. This table also has the frequency of each bigram for each drug. There exists a fixed list of all the unigram and bigram side effects collected from different medical websites. This list is used as a reference and words from the user reviews are picked up and inserted into this table.

The structure of the table is as follows:

**DrugName varchar(200)**: This is the field with the name of the drug.

**Bigram varchar(200)**: This field holds the bigram side effect itself (a bigram which appears in the list as well as in the user review).

**Frequency int(50)**: Frequency stands for no. of times a particular bigram has been mentioned as a side effect in context of a particular drug. The noteworthy point here is, any bigram that appears multiple times in the same review is still counted as 1 as it is important to know how many users have reported the bigram rather than how many times it appears throughout.

- **tabledsf** This table is very similar to the bitable described above. Just like bigrams, this table holds unigram side effects. Otherwise the logic is the same in the construction of both the tables.

**DrugName varchar(200)**: This is the field with the name of the drug.

**Effect varchar(200)**: This field holds the unigram side effect itself (a unigram which appears in the list as well as in the user review).

**Frequency int(50)**: Frequency stands for no. of times a particular unigram has been mentioned as a side effect in context of a particular drug.

- **thread** Other than these above mentioned tables which form the crux of the mining process, there are a few tables which support the forums feature of the website. One such table is the thread table with the following details:
username  varchar(20) : This is the field which has the username of the user who posts the review.

threadname  varchar(100) : This field is the topic of the thread in which the user posts a reply.

Data - varchar(1000) : Users signed in can post a reply to any thread or create their own thread.

• userdb  The last table is the Users database table maintaining the record of all usernames and passwords to facilitate login - logout.

username  varchar(20) : This field holds the unique user id.

password  varchar(15) : As the name suggests, password corresponding to the username.

fullname  varchar(50) : This field is just for the sake of information of the user.

Email  varchar(100) : Holds the email address of the user.

Sessid  varchar(50) : This holds the sessionID generated for each session. When the user logs out, this field is updated to hold NULL.

A.4 Implementation Details

This section provides a detailed description of the implementation structure, giving a file-by-file view of the entire architecture. This helps in easy understanding as well as facilitates easy modifications.

A.4.1 Words.pl

After all the pages have been extracted from the website as html files, they are parsed to generate text files with all the html tags removed. Then the reviews for all the drugs extracted from this website are inserted into the table called xyz.
Now the next step is to iterate through all these reviews to extract the medical terms/side effects, which are used for calculating frequency and other numerical calculations.

The file named words.pl is used to extract the medical terms from the user reviews given in database table xyz. The module fetches all the rows of (drugname, review) from the xyz table and iterates over every row processing the review to extract medical terms/side effects. The first step involves removal of words that do not add any semantic value to the sentence. For this purpose, there is a stop list (%stop_list) of words, which is used to eliminate the common words of English, which are definitely deemed not medical.

There is another list of words (%unigram), which are a collection of medical side effects and symptoms collected from the Internet by crawling various websites. This list is used as a reference against each review to bring out the drug to side effect mapping for every drug. A drug to side effect mapping in a particular review is counted only once even though it might contain multiple occurrences of the same mapping over and over again.

Only unigram and bigram side effects are considered for the present situation.

- The mapping with unigrams in them are stored in the table tabledsf.
- The mapping with bigrams in them are stored in the table bigram.

This mapping is used in the interface to bring out the reviews based on the users query of a combination of drug and symptoms. Now to look at the methodology:

For each review, the first step is to remove all the punctuation marks, convert all letters to lower case and remove leading spaces. This is done using the regular expressions feature of Perl programming language. The next loop considers every word of this review to check if it a stop-list word or not. If it is, then it is ignored. If not, then the first sub-step checks to see if this word exists in the unigram list.
If it does, then the next check is to see if the word already exists in the table. If it does, then it is ignored else, this word is inserted into the tabledsf table with the frequency as 1. The second sub-step is the exact same procedure for bigrams to insert into the bitable. The only difference here is, every two consecutive words are considered instead for single words. The above procedure takes place every time a new Drug row appears into consideration.

Other than the first occurrence of a Drug, all other rows are handled a little differently. The initial clean up and pre-processing remains the same. The difference is, every word is checked to see if it has been already inserted from any previous review. If yes, then the second check is to see if this word has already appeared in the same review. If yes, then it is ignored. If no, then the frequency is updated by incrementing by 1. The same procedure takes place with bigrams as well although two consecutive words are considered at a time.

### A.4.2 Links.php

Given the list of drugs in the pages indexed by alphabets, the goal is to display the side effects that each of the drugs exhibit and also give a distribution of the reported side effects based on various factors. For this the file links.php is used to give the desired result. As seen in the previous sections, all the reviews are present in the database table, xyz. Also the drug to side effect mapping along with their frequencies are present in the tables tabledsf and bitable.

When a user clicks on one of the drug names that are indexed in the alphabetical order (in the pages titled, list.php ( runs from a-z)), the name of the drug is sent as a php request to links.php. The first of links.php task is to get the top side effects that users have reported. This can be achieved by running a query on the tables tabledsf and bitable. The drug name is captured by links.php when the php request is sent and it is used to retrieve the side effects from the above-mentioned
tables for the drug in context. This is done as follows:

- SELECT distinct Effect, Frequency FROM tablesdf WHERE DrugName="".$drugg.""  
  ORDER by Frequency desc

- SELECT distinct Bigram, Frequency FROM bitable WHERE DrugName="".$drugg.""  
  ORDER by Frequency desc

- SELECT count(*) as ct FROM xyz where DrugName="".$drugg.""

Now that the drug to side effect mapping is ordered according to the frequency, the top 20 side effects among the unigrams and bigrams is brought out by comparing the frequencies of both the unigrams and bigrams. The frequencies considered for the purpose of populating the tables tablesdsf and bitable are not percentages. They are whole numbers. Therefore in order to show their percentage distribution, there is a need to calculate the frequency of a particular side effect over the total number of drugs. This is achieved first, by calculating the total number of reviews available for a particular drug (query 3 above). Then using this value, the calculation of the total frequency for a particular side effect is achieved by dividing the total number of reviews that contain the particular side effect by the total number of reviews for that drug ($ct).

Since the top 20 side effects are considered, for certain drugs, there is a chance of the number of side effects being less than 10 each. In that situation, a count for the number of unigrams is maintained ($count1) and a count of the number of bigrams is also maintained ($count2). This count is necessary, so that the side effects are displayed even if either of the unigrams and bigrams are less than 10 each.

The top 20 side effects for the drug are displayed in the right hand side division of the web page. Each of these side effects is displayed as a link, the reason being the goal of this file. The goal is to allow the user to interact with the web page. So the
side effects displayed are links to the reviews that have reported that particular side effect for the drug in context. This achieved by the AJAX functionality built into the web page. The function doWork() is used to achieve this. doWork() sends a php request to a file new.php which receives the tuple (drugname, sideeffect) and uses the onreadystatechange() function of AJAX to await the reply from the new.php page. The php reply from the new.php is redirected into the main div (content area) that exists on the left side of the page. Here the page is populated with the reviews of the side effect which the user clicks on for the drug in context.

The other functionality that is built in is the flexibility to see all the side effects that the drug exhibits. This is achieved by the usage of the function doWork1(). This uses the same redirect methodology as above but a different file to process named side.php and displays the entire list of side effects that the users of this particular drug have reported. side.php is sent a php request from the existing links.php page with the tuple (drugname). doWork1() uses the same onreadystatechange() function to await the reply and populates the div with a list of all the side effects for the drug in context.

The final functionality is that of the distributions given. The distributions are given on a pie chart of the frequency of top side effects vs the frequency of the total number of side effects displayed. Also a distribution of the top side effects is given on a bar graph where the frequency of the each side effect reported for this particular is compared with the frequency of the same side effects in the whole data base.

A.4.3 New.php

New.php receives a php request from either links.php or case2.php files with the tuple (drugname, side effect). This is received in the variable $q. $q receives the tuple as a single word separated by the special character . The split function is
used to separate the drugname from the side effect in the word stored in $q. The array $r now contains the drugname and the side effect which are assigned to the variables $dru and $sid respectively. The following query is used to bring out the reviews present in the database for the combination of this particular drug and side effect in context. The query is as follows:

- SELECT user, review, link FROM xyz WHERE drugname="'".$dru."'" and review like "'".$sid."'"'

Here the username, review and link are retrieved from the database table xyz and are assigned to the parallel arrays $user, $link, $reviews. The array $user contains all the user names of the people who have written reviews for the drug in context and have reported the side effect mentioned. The array $link contains the links to the web pages which contains the reviews of people reporting the side effect for the drug in context. The array $reviews contains the reviews written by the people for the particular (drugname, side effect) tuple. A count of the total number of reviews for the tuple (drugname, side effect) is maintained. This is used to determine whether the total number of reviews is greater than 10 or less than 10. If the total number of reviews is greater than 10, then only 10 reviews at random are presented to the user in the php response. If the total number of reviews is less than 10, then all the reviews are presented to the user in the php response. This is how the goal of presenting the reviews that users have a given for a particular drug based on the side effect that the user wants.

A.4.4 Side.php

Side.php receives a php request either from links.php or from case2.php files with the tuple (drugname). This is received in variable $q. $q contains the drug name of the drug in context and the goal of side.php is to respond with all the side effects that the drugs users have reported and its frequency. This is achieved by using
the following 2 queries:

- SELECT distinct Effect, Frequency FROM tabledsf WHERE DrugName="".$q.""
  ORDER by Frequency desc

- SELECT distinct Bigram, Frequency FROM bitable WHERE DrugName="".$q.""
  ORDER by Frequency desc

The first thing to do is to record the number of side effects recorded by retrieving the unigrams and bigrams from the database tables tabledsf and bitable. Then the side effects are echoed back as a php response to either links.php or case2.php. The frequency of the side effects is also reported as the ratio of each side effect reported for the drug in context ($ct1) to that reported over the entire database of drugs ($ct2).

A.4.5 Forums.php

The forums.php page is provided to users of this website to interact and share their opinions. In order to interact on the forums page, a user has to be a registered member of this web site. Therefore, a check is made immediately by retrieving the session id, using the session_id() function of php. This session id is checked against the registered user database table userdb to check if any user is already logged in. When a user logs in, the session id is recorded in the database table userdb. This determines whether a user is allowed to interact in the forums or can only view the forums. If the user is not logged in, then the user is allowed to view the threads but is not allowed to reply to any of the posts.

If the user is logged in, then the threads are displayed to the user to allow viewing of the various discussions present. The threads are displayed by retrieving the name of the threads and the number of replies given for that particular post:

- SELECT threadname,count(data) as k from thread group by threadname
The threadnames are displayed as links. By placing the links in a form (name=formstart) the javascript function getsupport() can be used to submit the form and with it send a php request to the file thread.php. The user when logged in also has the utility to create a thread and post comments in it. For this the button start new thread is placed in a form and the form action redirects to a file threadaddform.php. This is how a user can interact with the Forums in the web site.

A.4.6 Thread.php

This file is used to show the information contained in a particular thread. The user interacts with the forums.php page and clicks on a thread name. When the thread name is clicked, the user is redirected to the thread.php page which populates the page with the user names and messages that each user has posted with reference to this particular thread. If the user is logged in, then the user is allowed to post messages with respect to the thread in context. This is done by checking whether the user is logged in or not:

- SELECT count(*) as ct from userdb where sessid like '%%%'.$id.'%%'

If the count ct in the above query turns out to be 1, then there is a user already registered with the website and is logged in at the moment. Therefore, the user is given complete freedom to interact with the website and post comments. The count is recorded in the variable $y. If the count is not 1, then there is no user logged with the present session id and so the button that allows the user to post comments is disabled. When the user is logged in, a message with the user name is displayed above the text box asking the user to post a message. Otherwise, the user is asked to log in to post the message. The messages are retrieved as follows:

- SELECT username, data from thread where threadname ="".$thread.""

Here each username and data is recorded in the arrays $user and $data. Thread.php
is only used to display the posts in a particular thread to the user.

A.4.7 Loginform.php, Log.php and Logout.php

This file is used by the user to log in to the website and only when logged in will be allowed to interact with the web site. The link to this page is available on any page of the web site. The user is given 2 options:

- Login with existing user id
- Register for new user id

If the user is already a registered user then the first form of the page can be used to login. When the user enters the required credentials and clicks on the Login button, a php request is sent to the log.php page. Log.php has only 2 functions. With credentials sent over, the user name and password are stored in the variables $user and $pass. These are checked for existence in the database table userdb:

- SELECT count(*) as ct from userdb where username='"'.$user.'"' and password = '"'.$pass.'"

If the count in the above query turns out to be one, then the user is registered with the web site and an update query is run. The update in this case on the table userdb is the session id which determines whether the user is logged in or not:

- UPDATE userdb set sessid = '"'.$sid.'" where username = '"'.$user.'"

If the count doesnot yield a value of 1, then the user is not present in the database and will have to register. In such a case, the user is redirected back to the loginform.php. Once the user logs in, the page is redirected to the forums.php page where the user can start interacting. If the user chooses to logout, a logout link appears right next to his user name on every page of the
page web site. Once the user clicks the link, an update query is run so as to remove the session id from the corresponding tuple in the database table userdb. The update is as follows:

- UPDATE userdb SET sessid = 'NULL' where username = "'.$user.'""

Therefore, all these files are put together in a folder titled www in the Apache Directory. In addition to these files, the templates and images for the webpages are included in the htdocs folder of the Apache Appserver.

A.5 Sentiment Analysis, Search and Ranking

These have been discussed in the thesis. The Sentiment Analysis algorithm has been described in 3 stages on pages 29 and 30. Please refer to them while designing the Algorithm. Also code snippets are provided in Figures 10, 11 and 12 for Zend LUCENE index creation and search. Also insight into how to design the how to design the ranking algorithm has been discussed in Section 3.3.

A.6 Conclusion

The details provided in this document would provide a very comprehensive view into the features, details and programming modules of the application. This would act as a good introductory document for developers looking to improve on the present architecture by adding more features. The internet is a vast ocean and can provide more and more data which can be exploited and harvested to the maximum possible extent. This effort is a stepping stone to one such useful application, largely motivated by the audience response.