Indian Institute of Technology, Bombay

M.Tech Dissertation

Different approaches to Sentiment Analysis: Topic modeling and Deep Semantics

Author:
Nikhilkumar Jadhav

Supervisor:
Dr. Pushpak Bhattacharya

A report submitted in partial fulfillment of the requirements for the degree of M-Tech in Computer Science

in the

Computer Science and Engineering Department

June 2014

Dissertation Approval

This dissertation entitled **"Different approaches to Sentiment Analysis: Topic modeling and Deep Semantics"**, submitted by **Nikhilkumar Jadhav (Roll No: 123050033)** is approved for the degree of **Master of Technology** in **Computer Science and Engineering** from **Indian Institute of Technology Bombay**.

Prof. Pushpak Bhattacharyya
Dept. of CSE, IIT Bombay
Supervisor

Dr. Saketh Nath J
Dept. of CSE, IIT Bombay
Internal Examiner

Dr. SasiKumar M
CDAC, Mumbai
External Examiner

Prof. Anirudha Joshi
IDC, IIT Bombay
Chairperson

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: 13th June, 2014 Nikhilkumar Jadhav

Place: IIT Bombay, Mumbai Roll No: 123050033

Acknowledgements

First of all, I would like to thank my guide Dr. Pushpak Bhattacharya for guiding me throughout this work. I would also like to thank my family and all my friends for being there all the time and providing me the necessary support.

I would also like to thank the members of the *Sentiment Analysis* group at *IIT, Bombay* for their valuable feedback and suggestions.

Abstract

Sentiment Analysis is mainly the classification of text into two classes viz. positive and negative. Joint sentiment and topic models have been used to tackle this classification problem. Despite having a hierarchical structure, these generative models have a bag of words assumption. Due to this fact, they tend to misclassify texts having sentiment in the form of phrases. *LDA* and it's extensions don't work properly with phrases. To tackle this situation, we propose an unsupervised approach to sentiment analysis using topical n-grams which have been shown to be effective with phrases. We train the topical n-grams model using two topics i.e., positive, and negative, list of positive and negative words, and rules to detect positive and negative phrases. New documents are then classified using this trained model. The system gives better results than the existing Joint Sentiment Topic model. We also propose an approach to generate list of positive and negative words using *LDA*. Another aspect of this research is to make use of deep semantics for sentiment analysis. Existing methods for sentiment analysis use supervised approaches which take into account all the subjective words and or phrases. Due to this, the fact that not all of these words and phrases actually contribute to the overall sentiment of the *text* is ignored. We propose an unsupervised rule-based approach using deep semantic processing to identify only relevant subjective terms. We generate a *UNL* graph for the input *text*. Rules are then applied on the graph to extract relevant terms. The sentiment expressed in these terms is used to figure out the overall sentiment of the *text*. Results on binary sentiment classification have shown promising results.

Contents

Abbreviations

ML	**M**achine **L**earning
NLP	**N**atural **L**anguage **P**rocessing
IR	**I**nformation **R**etrieval
NB	**N**aive **B**ayes
SVM	**S**upport **V**ector **M**achines
SA	**S**entiment **A**nalysis
DCM	**D**irichlet **C**omponent distribution **M**odel
LDA	**L**atent **D**irichlet **A**llocation
LSA	**L**atent **S**emantic **A**nalysis
DAG	**D**irected **A**cyclic **G**raph
MCMC	**M**arkov-**C**hain **M**onte **C**arlo
UNL	**U**niversal **N**etworking **L**anguage

Chapter 1

Introduction

Sentiment Analysis is the technique of detecting sentiment/opinion behind a *word, sentence, collection of sentences, documents,* and even a *collection of documents* in some cases. Here, *word, sentence, collection of sentences, documents,* and *collection of documents* can be termed as chunks of text which determines granularity of the analysis. We can make use of the general term *text* when the discussion applies to all these chunks and specify the exact granularity when required. *Sentiment Analysis* might also involve classifying *text* as either *Objective* (factual information) or *Subjective* (expressing some sentiment or opinion). This is called as *Subjectivity Analysis*. It can also be considered as preprocessing for *Sentiment Analysis* in some cases. But *Subjectivity Analysis* is considered a task within *Sentiment Analysis*. *Sentiment analysis* is also known as *Opinion Mining* and these two terms are used quite interchangeably.

In most cases, *Sentiment Analysis* is a binary classification task in which a *text* is classified as either positive or negative. Examples of binary classification are *movie reviews, product reviews, etc. Ternary Classification*, wherein the *text* is classified as positive, negative or objective also has many applications.

This field is considerably new and is gaining of lot of attention. *Movie reviews* of critics are classified as positive or negative by using this technique. Same is the case with product reviews. *tweets, comments, etc.* are analyzed to detect the positive or negative sentiment behind them and sentiment about a particular entity. *IR* also makes use of *SA* these days to filter out subjective information and retrieve only the objective data. There is also a motivation for sentiment aware IR in which documents of relevant sentiment (either positive or negative) are fetched.

1.1 Motivation

The *Motivation* behind this research is to study sentiment analysis in general and the how can some of the techniques used in information retrieval be applied to it in particular. The unrealistic assumption made before applying this technique in many cases is that subjective data is available. Subjective text has to be retrieved from the web. Also, most of the data available is domain dependent. This makes most of the supervised methods not feasible for domain independent sentiment analysis. Thus, there is a need for a semi-supervised or unsupervised method which is domain independent. The fact that most of the previous works take into account all the sentiment-bearing words and phrases is the motivation behind using deep semantics.

1.2 Problem Statement

The aim of this project is to use different approaches like topic modeling and deep semantics for sentiment analysis. To achieve the first goal different topic models need to be used for *SA* and compared with each other on some standard data. To achieve the second goal, some sort of deep semantic processing has to be performed. This semantic information is then to be used to perform sentiment classification.

1.3 Contributions

Following contributions were made during the course of this project.

1. Studied *LDA*, *JST*, and *Topical n-grams* models.
2. Evaluated *LDA* for document clustering.
3. Used LDA and Topical n-gram model for binary sentiment classification.
4. Evaluated *LDA, JST,* and *Topical n-gram model* for binary sentiment classification task.
5. Proposed an approach for generation of positive and negative word lists using *LDA*.
6. Developed a rule-based system for ternary sentiment classification using *UNL*.

1.4 Roadmap

This report gives the problem statement and the contributions in Chapter 1. Chapter 2 starts with a psychological viewpoint of *sentiment*. This is followed by formal problem definition, and

then types of *SA*, challenges in *SA*, and some applications of *SA* are listed. Chapter 2 also explains basic machine learning techniques and their applications in *SA*. Chapter 3 starts with basics of *IR*. Then it moves on to discuss the various *text modeling* approaches prevalent in *IR* with a focus on *LDA*. It also shows how *SA* can be used for sentiment aware *IR*. Chapter 4 starts with the use of *LDA* for sentiment analysis. It then discusses two models which combine sentiment and topics. The pros and cons of these models have been discussed. Also, the topical n-grams models is explained and a technique to use it for the sentiment classification task is explained. Chapter 5 focuses on the second aspect of this research which is make use of deep semantics for sentiment analysis. . Chapter 6 concludes and hints on some future work.

SUMMARY

In this chapter, the problem statement was introduced along with the motivation. The significant contributions done so far have been listed. The structure of the report was described in the Section 1.4.

In the next chapter we will learn *Sentiment Analysis* and several machine learning techniques and how these are used in *Sentiment Analysis*.

Chapter 2

Sentiment Analysis

2.1 Sentiment: A psychological viewpoint

Though emotion is a term used often, it has a very complex psychological background. Emotion can be described as a component process. There are five organismic subsystems in an individual *viz. Information Processing, Support, Executive, Action, and Monitor* [Scherer, Bänziger, and Roesch, 2010]. Like any system, these subsystems also have states and they keep on transiting between different states according to the environment. Individuals respond to stimuli in the environment. Whenever an individual encounters a stimuli, state of these subsystems change and these changes are both interrelated and synchronized. This episode is called an emotion [Scherer, 2005].

Affect is the feeling or emotion experienced during or after the process of responding to a stimuli. This state of experience is called as *Affective State*. Thus, emotion is one of the affective states. *Mood* also can be considered as one of the affective states. *Attitude*, is one of the most important affective states. *Attitude* can be defined as *" enduring, affectively colored beliefs, dispositions towards objects or persons"*[Scherer, 2005]. *Sentiment Analysis* is the nothing but the detection of attitude towards an entity. Intuitively, we can see that it is possible to infer even the emotion. We can determine whether the user is sad, happy, angry,*etc,* if we know the attitude of the user.

2.2 Formal Problem Definition for Sentiment Analysis

Before devising any solution to a problem, it is advisable to have concise definition of the problem first.

2.2.1 Example

Let us consider an example to define the problem,

"1)I went to watch the new James Bond flick, ***Skyfall*** *at IMAX which is the best theater in Mumbai with my brother a month ago. 2)I really liked the seating arrangement over there. 3)The screenplay was superb and kept me guessing till the end. 4) My brother doesn't like the hospitality in the theater even now. 5) The movie is really good and the best bond flick ever"*

This is a snippet of the review for a movie named **Skyfall**. There are many entities and opinions expressed in it. 1) is an objective statement. 2) is subjective but is intended for the theater and not the movie. 3) is a positive statement about the screenplay which is an important aspect of the movie. 4) is a subjective statement but is made by the author's brother and also it is about the hospitality in the theater and not the movie or any of its aspects. 5) reflects a positive view of the movie for the author.

We can see from this example that not only the opinion but the opinion holder and the entity about which the opinion has been expressed are also very important for overall SA. Also, as can be seen from 1),4) and 5) there is also a notion of time associated with every sentiment expressed.

2.2.2 Problem Definition

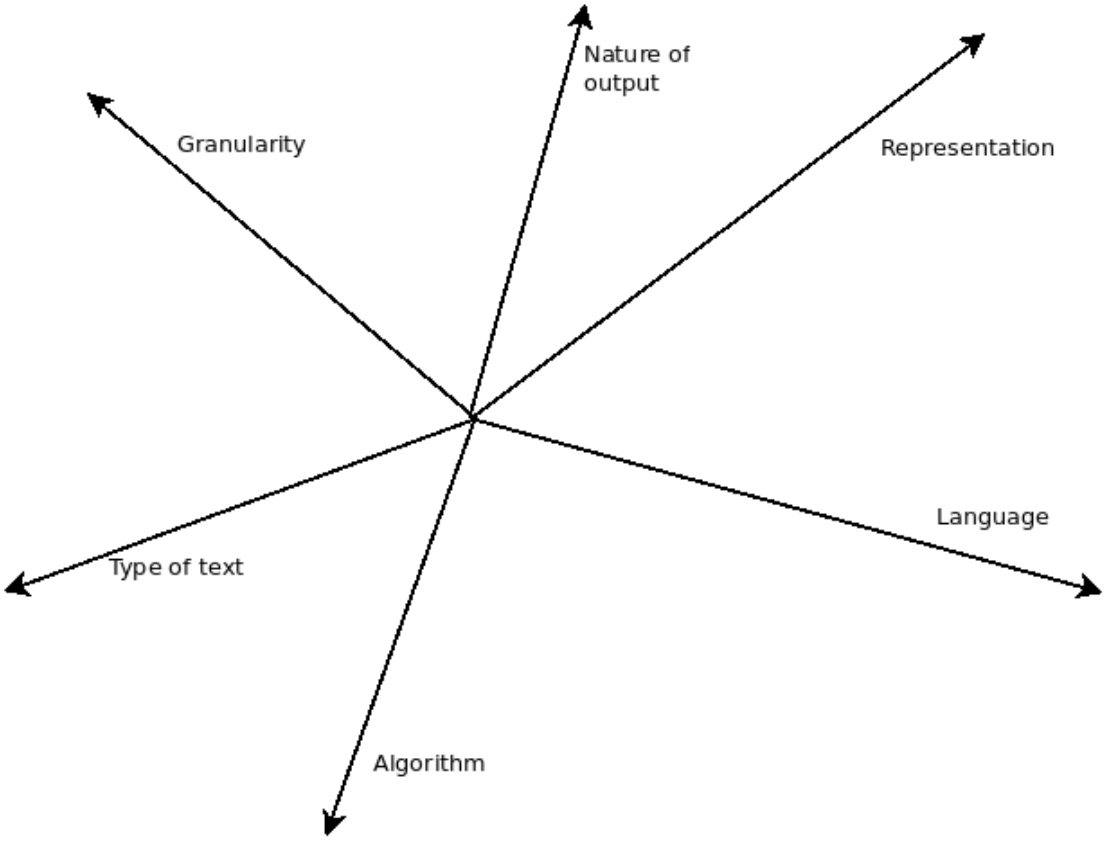
"A direct opinion (opinion about the object) is a quintuple $(o_j, f_{jk}, oo_{ijkl}, h_i, t_l)$*, where* o_j *is an object,* f_{jk} *is a feature of the object* o_j*,* oo_{ijkl} *is the orientation or polarity of the opinion on feature* f_{jk} *of object* o_j*,* h_i *is the opinion holder and* t_l *is the time when the opinion is expressed by* h_i *"*[Liu, 2010].

Thus we can see that the definition of sentiment analysis takes into account not only the object, opinion, and opinion holder but also the time and the specific feature about the sentiment is being expressed. This definition plays a very important role in devising any approach to solve any problem related to sentiment analysis.

2.3 Types of Sentiment Analysis

Sentiment analysis is primarily a classification task. But, we can also classify the task of sentiment analysis depending upon various features. These features or dimensions are shown in figure 2.1,

FIGURE 2.1: Dimensions of a Sentiment Analysis problem



1. Granularity of Text
2. Type of text
3. Algorithm
4. Language
5. Representation
6. Nature of Output

All these features collectively characterize a particular problem in SA. A change in even one of the features will change the problem.

2.4 Challenges

SA is a complex problem and has many challenges involved. In this section, an attempt is made to discuss some of the most notorious difficulties in *SA*.

2.4.1 Unstructured Text

Text in micro-blogs, tweets, comments, and messages is unstructured. Most of the research in *NLP* and many *NLP* tools focus on structured data. To adapt and use these tools for *SA* is a big challenge.

2.4.2 Sarcasm

Nowadays, many tweets and comments are sarcastic. Let us see an example on tweet, *"Great! I ate too many chocolates and gained lot of weight :)"*. This sentence will be marked as positive by almost any classifier. But, we can clearly see that this is not a positive statement. Correctly, classifying such sentences will require context knowledge.

2.4.3 Thwarting

In a thwarted expression, the sentences which contradict the overall polarity of the document are in majority. An example is, *"The guy is a chronic drinker, he smokes weed, has drugs but is a good guy"*. The aim of thwarted expressions is to mislead the classifier. Detecting thwarted expressions is a very important difficulty.

2.5 Applications of Sentiment Analysis

We list some of the most important applications of *SA*.

1. Classification of Tweets
2. Classification of Movie Reviews
3. Classification of Product Reviews
4. Analyzing market trends
5. Sentiment Aware Information Retrieval
6. Removing subjective sentences to improve IR performance

As we have discussed earlier, *SA* is mainly a classification task. It can be thought of as a subset of a another important classification task called *Text classification*. Various approaches have been used to solve this problem. Most of them are machine learning based. This chapter takes a look at basic machine learning and some models in brief. Later on, combination of these models with other techniques which take into consideration the various aspects of the *text* are considered.

Approaches to Sentiment Analysis

2.6 Machine Learning

This section aims not to explain all the intricacies associated with machine learning but to provide a brief outline which suffices for understanding future concepts. Machine learning by definition is learning from data. We have a mathematical model, parameters of which have to be estimated from the data. By doing this we fit the model to the data provided. These parameters which have been *learned* from the data can be said to completely define the model. Now this *learned* model can be used for prediction or classification. In the case of *SA*, *Machine Learning* is used for classification of *text* as either positive, negative or neutral.

The data available to us can be both labeled and unlabeled. By labeling we mean that the for an instance of the input we know its class. Data labeling can also be called annotation. Labeled data is called annotated data. Depending upon the extent to which the training data is labeled, we can classify the *ML* techniques as follows.

2.6.1 Supervised

In this case, all the training data is a labeled. Majority of ML based techniques have this requirement that the data should be completely labeled. The accuracy of the system decreases if the data is very small in size. This is called data sparsity problem. They tend to over-fit if the data size is small.

2.6.2 Unsupervised

Here, the data is completely unlabeled. As opposed to supervised techniques, they do not suffer from data sparsity problem as the input in unlabeled.

2.6.3 Semi-supervised

In this we have a mixture of both labeled and unlabeled data. Using the labeled data we try to annotate the unlabeled data. This technique is very useful if we have very sparse data.

2.7 Feature Vector

Feature vector can be thought of as a way to represent the input. Some important aspects of the input are considered and used to represent it in the form of a vector of values. These values contain some important information which aids the classification algorithm.

2.8 Models used for classification

When we say we learn from the data, we actually train the model to do so. There are broadly two ways to do this. One is in a generative way and the other is discriminative. What do we actually mean by this? We want to infer the class of a *text*.

Let x represent the input *text*. We want to determine the class y given the input *i.e*, we are interested in modeling $p(y \mid x)$. There are three ways to do this.

2.8.1 Generative Models

One way is to model p(x, y) directly. Once we do that, we can obtain $p(y \mid x)$ by simply conditioning on x. And we can then use decision theory to determine class membership *i.e*, we

can use loss matrix, *etc.* to determine which class the point belongs to (such an assignment would minimize the expected loss). We can learn p(y), the prior class probabilities from the data. We can also learn $p(x \mid y)$ from the data using say maximum likelihood estimation (or we can Bayes estimator, if you will). Once you have p(y) and $p(x \mid y)$, p(x, y) is not difficult to find out.

Bayes' rule is given as follows,

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)} \tag{2.1}$$

2.8.2 Discriminative Models

Instead of modeling p(x, y), we can directly model $p(y \mid x)$, for *e.g*, in logistic regression $p(y \mid x)$ is assumed to be of the form,

$$p(y \mid x) = \frac{1}{1 + \mathrm{e}^{(-\Sigma(wi.xi))}} \tag{2.2}$$

All we have to do in such a case is to learn weights that would minimize the squared loss.

There is a very easy technique to tell whether a model is generative or discriminative. If we can a generate new training data using the model then it is certainly generative. In the case of generative models, the distribution $p(x \mid y)$ is a model which fits the training data so it can be used to generate new data. In the case of discriminative models, this is not the case.

2.8.3 Encoding a function

We find a function f(.) that directly maps x to a class. The best example of this is decision trees.

2.9 Models

Here, we explain the various machine learning models used.

2.9.1 Naive Bayes classifier

Naive Bayes classifier is a generative classifier with strong independence assumptions. This means that all the features in feature vector are independent of each other given the class. Despite of

this very naive assumption, it gives surprisingly good results. The parameters are estimated using the method of maximum likelihood.

Suppose we want to determine value of the class variable C of the given input consisting of feature variables $F_1,\ldots,F_n$, it can be expressed as a conditioned probability $p(C \mid F_1,\ldots,F_n)$ using the Bayes' theorem,

$$p(C \mid F_1,\ldots,F_n) = \frac{p(C)p(F_1,\ldots,F_n \mid C)}{p(F_1,\ldots,F_n)} \tag{2.3}$$

To use this equation for classification of a given input x which is represented by a feature vector $(F_1{=}f_1,\ldots,F_n{=}f_n)$, the following equation is used,

$$class(f_1,\ldots,f_n) = \arg\max_c p(C=c)\prod_{i=1}^{n} p(F_i = f_i \mid C=c) \tag{2.4}$$

The main advantage of Naive Bayes is that it gives remarkably high accuracy for relatively small data sets because of the independence assumption.

2.9.2 Maximum Entropy

Maximum Entropy more commonly known as *MaxEnt* is a discriminative classifier. As it is a discriminative classifier, here we find out the conditional distribution of the class variable. The basic principle underlying maximum entropy is that without external knowledge one should prefer distributions which are uniform and have maximum entropy. Using the training data, constraints on the distribution are derived which can be used to infer where the distribution should be minimally non-uniform. These constraints represent the expected values of the features [Nigam, Lafferty, and McCallum, 1999]. In Text classification, *MaxEnt* estimates the conditional distribution of the class label given the *text*. Representation of the *text* is mostly in terms of word count features or word presence features.

Let f be the features that link the observation x to the class c. A feature in this case is a function denoted by $f_i(x,c)$ with a bounded real value. Also, let X denote the collection of *texts*. The aim of *MaxEnt* is to restrict the model distribution to have the same expected value for each such feature as is seen in the training data X. This means that the learned conditional distribution $p(c \mid x)$ must satisfy the following property,

$$\frac{1}{|X|}\sum_{x\in X} f_i(x,c(x)) = \sum_x p(x)\sum_c p(c \mid x) f_i(x,c) \tag{2.5}$$

equation 2.5 reduces to the following form as we are not interested in modeling the collection X here.

$$\frac{1}{|X|}\sum_{x\in X} f_i(x, c(x)) = \frac{1}{|X|}\sum_{x\in X}\sum_{c} p(c \mid x) f_i(x, c) \tag{2.6}$$

Feature identification is very important in this case. Using the training data, expected value of features is used as a constraint for the model distribution. A class label for which most of these constraints are satisfied is the class of the given input x.

2.9.3 SVM

A basic *SVM* is a non-probabilistic binary linear classifier which given an input data predicts which of the two possible classes forms the output.

2.10 Usage in SA

We have covered lot of basics to easily understand some of the techniques used for SA. We start with a basic bag of words model and then move on to more advanced techniques which incorporate the attributes of *text*. Discourse based technique is discussed followed by a technique which makes use of minimum cut of a graph. These are followed by an unsupervised method which makes use of a search engine called *Alta Vista* to determine the semantic orientation of the *text*. The last method we discuss is a semi-supervised method which aims to perform ternary classification of sentences.

2.10.1 Bag of Words

In a Bag of Words model, the feature vector is just a unigram model which is used to represent the presence or absence of a word. Let us consider an example sentence, *"I hate to play football"* and suppose the vocabulary V is $\{I, You, like, hate, to, for, play, dance, football, cricket\}$. In this case, the feature vector will be *(1,0,0,1,1,0,1,0,1,0)*. We can see that here every word is a feature. In addition, it makes use of list of positive and negative words. If a word is positive then the value corresponding to that feature is +1 and if it is negative the value is -1. Thus for the example is our case, the feature vector after making use of this dictionary becomes *(1,0,0,-1,1,0,1,0,1,0)*. This feature vector is nothing but a representation of the input. If sum of all the values is positive then the sentence has positive polarity and if it is less than zero then it has

TABLE 2.1: Classifier Accuracy

Features	Keywords	Naive Bayes	MaxEnt	SVM
Unigram	65.2	81.3	80.5	82.2
Bigram	N/A	81.6	79.1	78.8
Unigram + Bigram	N/A	82.7	83.0	81.6
Unigram + POS	N/A	79.9	79.9	81.9

negative polarity. For our example, it turns out that the sentence is negative. The accuracy of such a system though is not very good, around 65 %.

On the other hand, If this input is fed to a default classifier like *NB*, *SVM* or *MaxEnt* then it is shown to have a considerable increase in accuracy[Go, Bhayani, and Huang, 2009]. In [Go, Bhayani, and Huang, 2009], they conducted various experiments and got the results as shown in table 2.1

2.10.2 Adding Discourse information

Discourse elements have a sentiment changing impact on sentences. Let us consider an example,

"The screenplay was good but I didn't like the movie"

Feature vectors discussed in the previous section won't be able to encode the information in such sentences. Using Bag of Words model can result in classification to a completely opposite polarity. To detect the polarity of such a sentence, detection of discourse elements and determining their effect is very important. Many approaches for this are present but all of them are for structured data and on most of the micro-blogging sites, the content is unstructured. Unstructured data is the main reason that many sentiment analysis tools make use of bag-of-words model.

To solve this problem, it is important to categorize the various types of discourse relations and find out the semantic operators which play a major role. In [Mukherjee, Bhattacharyya, and Balamurali, 2010], discourse relations have been categorized and many examples have been given to emphasize on some specific relation. Also, the semantic operations influencing the polarity have been explained. The algorithm then takes into consideration all these attributes to create a feature vector. A weight is assigned to each valence shifter taking into consideration its position w.r.t the discourse element. Also, the polarity/sense of a particular word is flipped depending upon its position. It also makes an attempt to take into account the impact of modals, which should lower the weight in some cases. The feature vector thus consists of weight, polarity, flipping and modality values for each word in the sentence. Words having zero weight have do not affect the polarity in any way and are thus ignored while calculation. The Feature vector

thus created can be used for calculation the sentiment behind the sentence. Two methods have been used, one is Lexicon based classification and the other is *SVM* classification. In the *SVM* classification, words with the same root are represented with a single vector. Also, special handling of *emoticons* is present. *WSD* is also used to determine the exact sense of each word. The results show that this algorithm outperforms all the methods by a margin which has statistically significant. Also, since this is lightweight and extends the baseline bag-of-words model, the performance of the system is very good.

2.10.3 Influence of Objective sentences on Classification

In [Pang and Lee, 2004], they have attempted to classify movie reviews as positive or negative. In this case the granularity of the *text* is a *document*. Movie reviews often contain description of the plot. This description might contain polar sentences but they have no relation whatsoever with the review about the movie. These sentences don't help in describing about how good or bad the movie is. Consider the following example sentence from the review of a recent movie,

"The action follows Jean Valjean, a former convict, as he flees Javert, a fanatical police officer, through a version of 19th-century France peopled with various grotesques, victims and tarnished saints."

As we can see this sentence has a number of words with negative polarity. This will be classified as a negative sentence which might lead to the review as a whole being classified as negative. But, we know that this sentence is from the description of the plot and is not an opinion/review about the movie. To solve this problem, we need to identify which sentences are objective and discard them. Subjective sentences in this scenario mean those sentences which are meant to describe the plot or contain some factual information.

The approach followed in [Pang and Lee, 2004] has three steps,

1. Label the sentences in the document subjective or objective
2. The objective sentences are discarded. This extraction is based on minimum cut formulation which integrates inter-sentence contextual information with Bag of Words model
3. A standard machine learning approach is applied to the extract for polarity classification

Subjectivity Detection

This is the first step in their approach. They try to identify subjective sentences. For this, they have made use of cut-based classification. Also, coherence of sentences is also taken into

consideration. Coherence means that subjectivity status of two sentences close to each other may be same.

"I really loved the screenplay of this one. Award Winning Directors are often good at making such movies"

As we can see from this example, two coherent sentences should ideally be classified under the same class.

Cut-based classification

Let $x_1, \ldots, x_n$ be the sentences in the *document*. Also, let $C_1, \ldots, C_k$ be the classes into which the sentences are to be classified. There are two important sources of information which can aid the classification.

1. Individual scores: $ind_i(x_i)$ - Preference of each x_i for being in class C_j
2. Association scores: $asso(x_i, x_j)$ - Estimate of how important it is that x_i and x_j are in the same class

So, in the cut based classification penalizes if tightly associated items are put in different classes. So, taking into consideration, the objective is to minimize the cost given below.

$$\sum_{x \in C_1} ind_2(x) + \sum_{x \in C_2} ind_1(x) + \sum_{x_i \in C_1, x_k \in C_2} assoc(x_i, x_k) \tag{2.7}$$

Now, we can see that this problem in intractable as there are 2^{n}possible partitions of $\mathrm{x_i}$'s. This minimization problem can be solved by building an undirected graph G with vertexes $v_1, v_2, v_3, \ldots, v_n, s, t$. The last two are source and the sink and represent the two classes

The graph consists of following edges

1. n edges (s, v_i) with weight $ind_1(x_i)$
2. n edges (t, v_i) with weight $ind_2(x_i)$

A cut (S, T) of G is a partition of its nodes into sets $S = s \cup S$ and $T = t \cup T$ where S does not contain s and T does not contain s. Its cost $cost(S, T)$ is the sum of the weights of all edges crossing from S to T. A minimum cut of G is one of minimum cost.

Polarity classification

Using the cut-based classification, we get the subjective sentences in the movie review. This extract is the fed to default polarity classifiers which in this case are *NB* and *SVM*. The feature vector used for them is unigram based.

Significance

A cleaner document can be obtained by extracting only the subjective information. The accuracy of the polarity classification improves as they don't get irrelevant data.The performance of the system will improve as it has less text to work on.

2.10.4 Unsupervised semantic orientation

A completely unsupervised approach to SA has been used in [Turney, 2002]. It is aimed for classification of reviews about products, automobiles, review, *etc.* The approach used has three steps.

1. Find phrases in the review containing adjectives and adverbs
2. Find semantic orientation of such phrases
3. Take avg of semantic orientation. If it is positive then *Thumbs up* else *Thumbs down*

Step 1

Extract phrases containing adjectives or adverbs. Adjectives are good indicators of subjectivity. Adjectives in isolation have insufficient context to determine semantic orientation. *e.g*, *unpredictable* when applied to *steering* in a car review has negative semantic orientation. On the other hand *unpredictable plot* in a movie review has positive orientation.

Therefore the algorithm uses 2 consecutive words where one is adjective/adverb and the other provides context.

Step 2

Point-wise Mutual Information (PMI)

Point-wise Mutual Information between two words word1 and word2 is defined as

$$PMI(word_1, word_2) = log_2\left[\frac{p(word_1 \wedge word_2)}{p(word_1)p(word_2}\right] \tag{2.8}$$

where $p(word_1 \wedge word_2)$ is the probability that $word_1$ and $word_2$ co-occur.

Semantic Orientation

Semantic orientation of a phrase is given by,

$$SO(phrase) = PMI(phrase, excellent) - PMI(phrase, poor) \tag{2.9}$$

The *PMI* are estimated by issuing queries to a search engine which explains the *IR* in *PMI-IR*. It notes the number of hits to calculate the probability values. The search engine used in this case was *AltaVista*.

After some algebraic simplifications and using the fact that $p(phrase, excellent) = hits(phrase\ NEAR\ excellent)$, equation 2.9 reduces to the following form:

$$SO(phrase) = log_2\left[\frac{hits(phrase\ NEAR\ excellent)hits(poor)}{hits(phrase\ NEAR\ poor)hits(excellent)}\right] \tag{2.10}$$

Here, *hits(query)* stands for the number of hits returned by the query.

Step 3

In this step, average of the semantic orientations of all the phrases is taken.

- If the avg is positive then Thumbs Up.
- If the avg is negative then Thumbs Down.

Observations

After experiments were conducted, following observations were made.

- Movie Reviews are hard to classify because a good movie might contain unpleasant scenes. Description of such scenes might decrease the semantic orientation.
- Accuracy did not increase just by accounting for this bias using a constant value. Just as positive reviews have description of unpleasant scenes, negative reviews might contain description of pleasant scenes.

Limitations

This approach has some limitations as can be seen from the architecture.

- Queries search engine to calculate semantic orientation of each phrase.
- Every phrase is given equal importance. There should be a weighted sum.
- The performance in case of movie reviews is not good because it does not take into account the fact that the whole is not necessarily a sum of parts as pointed out in section. 2.10.3

2.10.5 Semi-supervised

A semi-supervised approach for detecting term orientation was used in [Esuli and Sebastiani, 2006]. In [Esuli and Sebastiani, 2006], they have tried to perform a ternary classification of terms as objective, positive, or negative. The motivation behind this study is that in most works which determine the term orientation, it is assumed that we already know whether it a subjective term or not. So, we assume that a lexical resource of subjective/objective terms is available. This is not the case.

Semi-supervised was discussed briefly in section 2.6.3. Semi-supervised can be depicted as shown in figure 2.1

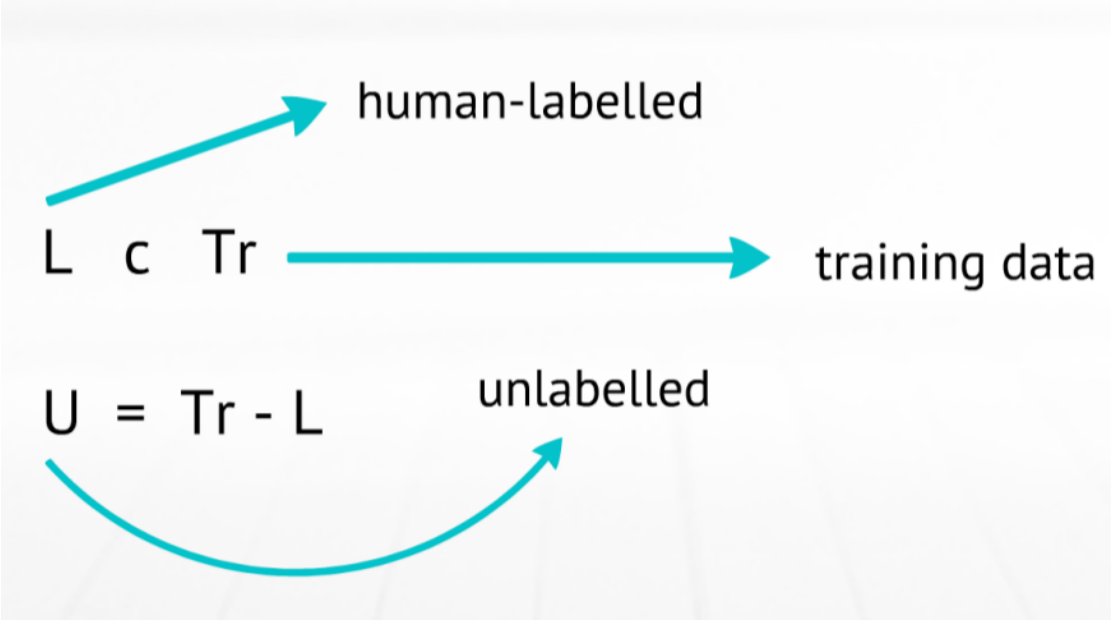


Figure 2.1 Training Data in semi-supervised learning

The unlabeled data has to be labeled and it should also be used for training. The labeled usually is called as seed set. In [Esuli and Sebastiani, 2006], they have made used of synonymy-antonymy relations of the wordnet to create the training data.

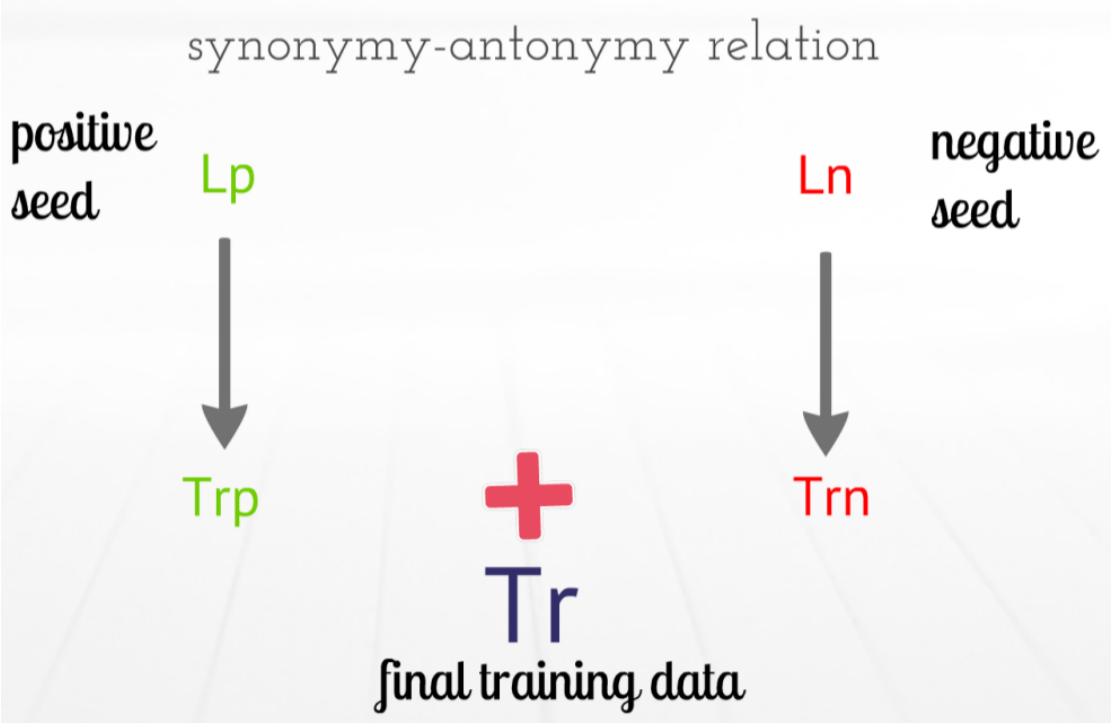


Figure 2.2 Using synonymy-antonymy relation for labeling unlabeled data

As we can see, we start with very small seed sets in this case, each containing only one element. The exact procedure is shown in the figure 2.3.

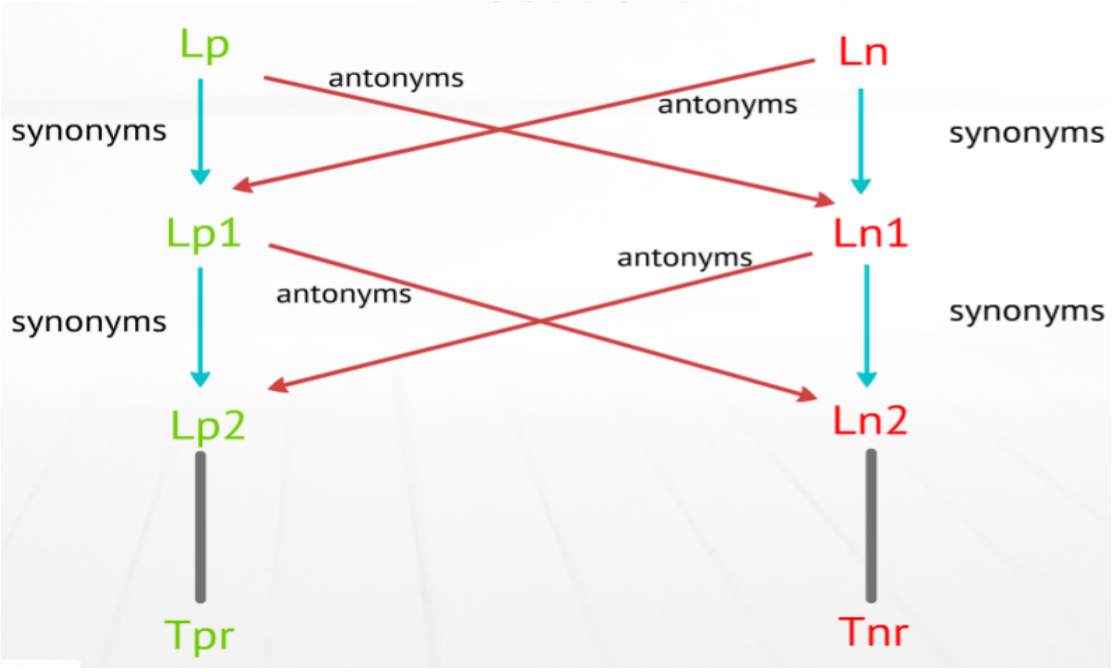


Figure 2.3 Procedure for labeling unlabeled data using synonymy-antonymy relation

After getting all the training data, a textual representation for each term is generated by collating all the wordnet glosses for that term. A cosine normalized *TF-IDF* representation is used as the feature vector. The result obtained by the approaches used in this paper were not that accurate and they show that algorithm used for term orientation when used for ternary classification perform badly and need improvement.

SUMMARY

In this chapter we introduced *Sentiment Analysis*. The motivation behind this work was discussed. A psychological viewpoint of *SA* was explained. A formal problem definition of sentiment analysis was given. This was followed by depicting the various dimensions of a problem in *SA*. Prevalent challenges in sentiment analysis were discussed briefly and some major applications were listed. Then we started with the basics of Machine Learning. Then we explained the various approaches and techniques used in ML. We explained what is a feature vector. Moving forward, we tried to understand the different models used in ML. Works using all these techniques were explained.

In the next chapter we will discuss information retrieval in brief and the focus on corpus models, mainly *LDA*.

Chapter 3

Corpus Models for Information Retrieval

In this chapter, we will explain information retrieval briefly. After that we will the various corpus models used for *IR*. The popular *LDA* model will be explained in detail later followed by it's evaluation. Let us first see what is information retrieval.

3.1 Information Retrieval

In simple terms Information Retrieval is the process of retrieving information relevant to the need. As the web contains lot of information, finding most relevant information is very difficult. A user usually requests information in the form of a query. The retrieval engine then presents the user set of documents relevant to the user's query. As the web contains lot of noise, it is difficult to fetch all the relevant documents. It should be noted that IR is not only concerned with the web as a resource of information. Precision, Recall, and F-measure are the important performance measures of an IR system. They are defined as follows.

Precision

Precision is the fraction of documents that are relevant to the query

$$Precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|} \tag{3.1}$$

Recall

Recall is the fraction of the documents that are relevant to the query that are successfully retrieved

$$Recall = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{relevant\ documents\}|} \tag{3.2}$$

F-measure

F-measure is the harmonic mean of *Precision* and *Recall*

$$F = \frac{2.Precision.Recall}{(Precision + Recall)} \tag{3.3}$$

Text indexing, relevance ranking, similarity search and corpus modeling are amongst the most important approaches used for *IR*. In the next section, we will show how sentiment analysis for sentiment-aware IR.

3.2 Sentiment Aware IR

In this section we will be discussing about two systems implemented to use *sentiment analysis* in *information retrieval*.

1. Indexing followed by Sentiment Analysis
2. Encoding sentiment in the Index

As can be seen by the brief description, the first one uses a staged approach and the second one is a one-stage process but involves some heavy preprocessing to predict the sentiment of the *text*. These approaches are very simple and aren't novel. These systems have merely been discussed to show the novelty of some future work. *Lucene* [Lucene, 2013] was used for indexing in both these systems.

3.2.1 Indexing followed by Sentiment Analysis

Architecture

1. Lucene

   It has a variety of features to perform indexing from basic to very advanced.

2. Query Processor

   This is also a part of of lucene. The objective content of the query is processed by this component

3. Sentiment Analyzer

   Sentiwordnet [SentiWordNet, 2013] was used to calculate the score of each word. The sentiment for a sentence was a sum of sentiment scores for each word. Sentiment of the whole document was a sum of sentiments for all the sentences.

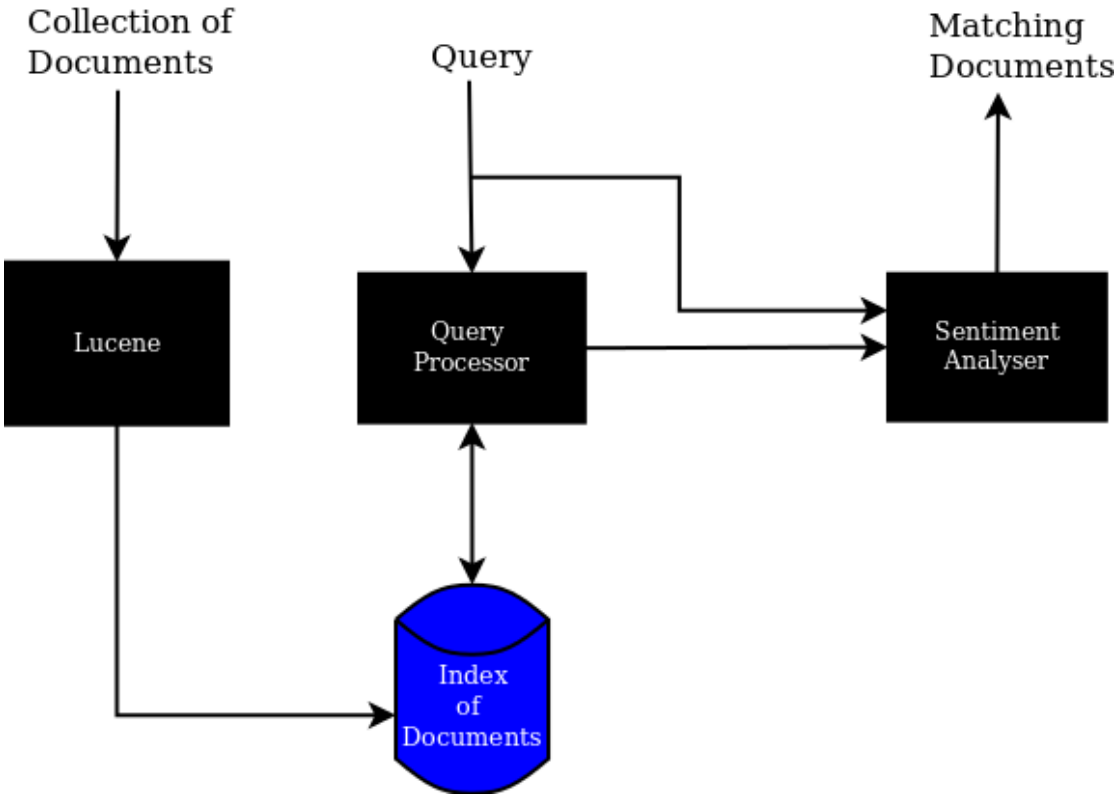


Figure 3.1 Indexing Followed by Sentiment Analysis

3.2.2 Encoding sentiment in the Index

Architecture

1. Lucene + Sentiment Analyzer

   It has a variety of features to perform indexing from basic to very advanced. In this case, sentiment analysis was combined with indexing. One more field, *sentiment* was added to the index. This value was inferred using the sentiment analyzer.

2. Query Processor

In this case, both the subjective and objective content of the query was processed. The documents were fetched based on the objective content of the query. Then, depending on the value of the *sentiment* field, these fetched documents were filtered and presented to the user.

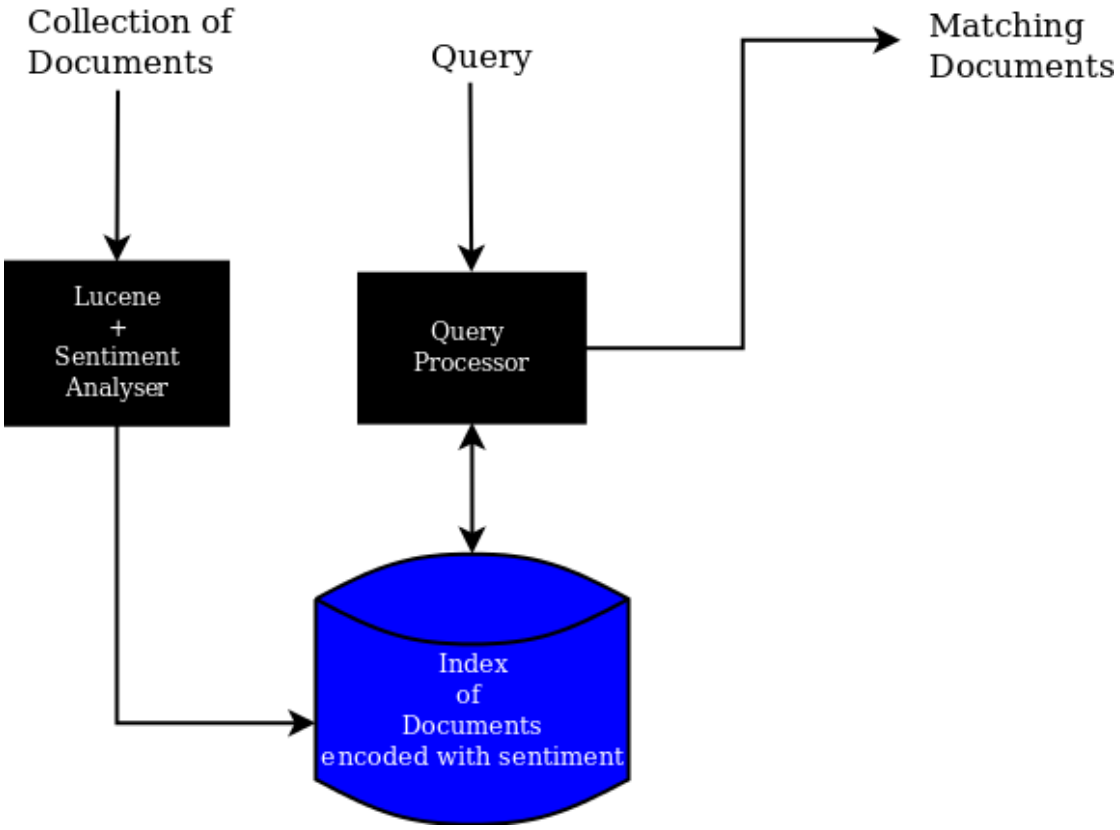


Figure 3.2 Encoding Sentiment in Index

Both these systems can be very useful. But as we can see the first one incurs lot of processing overhead and the second one has lot of preprocessing overhead. Also, both follow a procedural approach. A more novel approach will be two combine sentiment and topic to create a model which we can fit to our training data and then infer the resemblance of a new document to existing ones.

In this chapter we will focus on corpus modeling as in most works combining *SA* and *IR*, corpus modeling has been used. This draws directly from the Language Modeling approach used in many *NLP* systems. In the next section we focus on the different types of Corpus models. One effort towards using language modeling for *IR* was made in [Ponte and Croft, 1998]

3.3 Corpus Models

Corpus models are the probabilistic language models using which we can perform retrieval. In this section, we will have a look at multivariate binary model, *poisson* model, *multinomial* model, *DCM*, *dirichlet* smoothed models, and *LDA*. One advantage of using models to perform text retrieval is that the models can be refined independent of the retrieval algorithm.

3.3.1 Multivariate Binary Model

A document in this model is represented as a bit vector where the presence or absence of a word in the vocabulary is represented by a bit. The probability of a document x is given by,

$$Pr(x \mid \phi) = \prod_{w \in W} {\phi_w}^{x_w} (1 - \phi_w)^{1 - x_w} \tag{3.4}$$

$$= \prod_{w \in W} \phi_w \prod_{w \in W, w \notin x} (1 - \phi_w) \tag{3.5}$$

w in equation 3.4 is a word in vocabulary W. This model does not work well for short documents because in that case $|W| \gg |x|$. Also, the product make strong independence assumptions leading to the underestimation of $Pr(x \mid \phi)$.

3.3.2 Poisson Model

In the previous model, word counts were not taken into consideration. For this model, the document will be represented by a vector of word counts. Every word w has a parameter μ_w associated with it. In this model, it is assumed that word counts are random variables X_w that follow *poisson* distributions with means μ_w as follows:

$$Pr(X_w = z) = \frac{e^{-\mu_w} {\mu_w}^z}{z!}, z = 0,1,2,\ldots \tag{3.6}$$

The probability of invoking the Poisson document generator and getting a count vector x is given as:

$$Pr(x \mid \mu) = \prod_{all\ w} Pr(X_w = x_w) \tag{3.7}$$

$$= \prod_{all\ w} \frac{e^{-\mu_w} {\mu_w}^{x_w}}{x_w!} \tag{3.8}$$

$$= exp(-\sum_{allw} \mu_w) \prod_{w \in x} \frac{{\mu_w}^{x_w}}{x_w!} \tag{3.9}$$

3.3.3 Multinomial Model

In the previous two models, we were not able to model the document length. In this model, we can take that aspect of the document into consideration. Let L be the random variable for the document length. The length of the document in sampled from the distribution of this variable. For each word w in the vocabulary W, a probability θ_w is present. Let x_w denote the count of word w and l_x denotes length of the document.

The probability of the document with length l_x is given by:

$$Pr(l_x, \{x_w\}) = Pr(L = l_x) Pr(\{x_w\} | l_x, \theta) \tag{3.10}$$

$$= Pr(L = l_x) \binom{l_x}{\{x_w\}} \prod_{w \in x} {\theta_x}^{x_w} \tag{3.11}$$

$$= Pr(L = l_x) l_x! \prod_{w \in x} \frac{(\theta_x^{x_w}}{x_w!} \tag{3.12}$$

The drawback with the models discussed so far is that do not handle unknown words. If the document contains a word w which is not in the vocabulary W then it's probability will be zero. The models we see next overcome this drawback. They also take into account the word *burstiness* which means that when a word appears once in a document, it tends to appear more.

3.3.4 Dirichlet distribution model

In this model too the document is represented as a vector of word counts. The problem with multinomial model is that it fails to account for word *burstiness*. The nature of data according to *Zipf's* law follows a model of the form $data^{parameter}$ but in case of multinomial it is $parameter^{data}$. It is this intuition that lead to look for new models for representing data.

Dirichlet distribution is a probability density function over distributions given as

$$p(\theta \mid \alpha) = \frac{\Gamma\left(\sum_{w=1}^{W} \alpha_w\right)}{\prod_{w=1}^{W} \Gamma(\alpha_w)} \prod_{w=1}^{W} {\theta_w}^{\alpha_w - 1} \tag{3.13}$$

Equation 3.13 can be written exponential family form as

$$logp(\theta \mid \alpha) = \sum_{w=1}^{W} (\alpha_w - 1) log\theta_w + log\Gamma(\sum_{w=1}^{W} \alpha_w) - \sum_{w=1}^{W} log\Gamma(\alpha_w) \tag{3.14}$$

In this model, the representation of the document is in terms of a probability vector.

Documents being sparse in nature *i.e*, each document contains only a small subset of the vocabulary which might make the probability zero. Smoothing can be used in this case.

3.3.5 DCM (Dirichlet Compound Multinomial Model)

The problem with using smoothing in *DCM* is that over-smoothing is done. In this case all the rare words have the same probability of appearing in all the classes. Since, rare words are the main discriminators in classification, this model is not suitable for classification [Madsen, Kauchak, and Elkan, 2005]. A hierarchical model can solve this problem. *DCM* is one such model. It was introduced in [Madsen, Kauchak, and Elkan, 2005].

To generate a document using the *DCM*, a sample is first drawn from the Dirichlet to get a multinomial distribution, then words are iteratively drawn for the document based on the multinomial distribution. *DCM* can be thought of as a bag-of-bag-of-words-model. [Madsen, Kauchak, and Elkan, 2005]

The probability of a document x is given by:

$$p(x \mid \alpha) = \int_{\theta} p(x \mid \theta) p(\theta \mid \alpha) \mathrm{d}x \tag{3.15}$$

3.3.6 Latent Dirichlet Allocation

All the models discussed previously were for a single topic. We now consider one multi-topic model, *LDA*. It was introduced in [Blei, Ng, and Jordan, 2003]. Latent Dirichlet allocation is a way of automatically discovering topics that documents contain.

In more detail, *LDA* represents documents as mixtures of topics that spit out words with certain probabilities. It assumes that documents are produced in the following fashion:

- When writing each document, you decide on the number of words N the document will have.
- Choose a topic mixture for the document using dirichlet hypergenerator.
- Generate each word in the document by:
    1. First picking a topic using the multinomial distribution generated from the dirichlet hypergenerator.
    2. Then using the topic to generate the word itself.

Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection. A detailed discussion of LDA can be found in the next section.

3.4 Latent Dirichlet Allocation

Latent Dirichlet Allocation is a probabilistic generative model. It is completely unsupervised in nature. The main goal of *LDA* is to do *Latent Semantic Analysis* (*LSA*). *LSA* aims to find the latent structure of topics or concepts within a *text*. [Heinrich, 2005] gives a very thorough explanation for *LDA* and the inference method used. Most of the content in this chapter has been borrowed from [Heinrich, 2005].

3.4.1 Bayesian Network for LDA

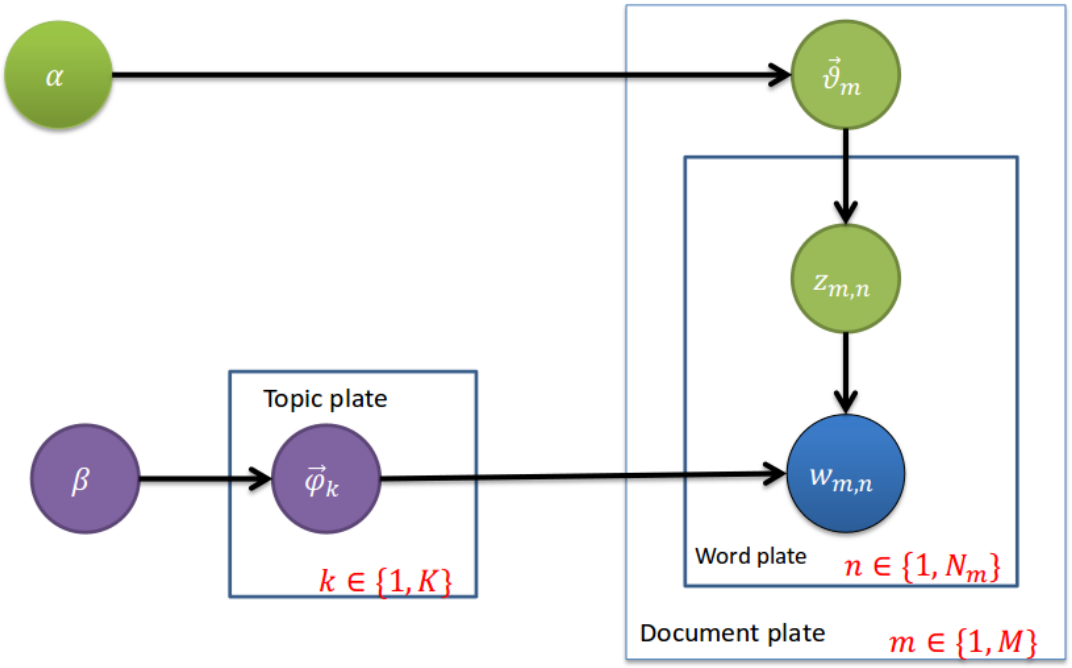


Figure 3.3 Bayesian Network for Latent Dirichlet Allocation

Bayesian Network is a *directed acyclic graph* (DAG) where nodes correspond to random variables and edges correspond to conditional probability distributions. The replication of a node is represented by a *plate*. This is to account for multiple values or mixture components.

3.4.2 Generative Model for LDA

The generative process for *LDA* is as follows :

☐ `Topic Plate :`
`for all topics` $k \in [1, K]$ `do`
  `sample mixture components` $\vec{\psi_k} \sim Dir(\vec{\beta})$
`end for`
☐ `Document Plate :`
`for all documents` $m \in [1, M]$ `do`
  `sample mixture proportion` $\vec{\vartheta_m} \sim Dir(\vec{\alpha})$
  `sample document length` $N_m \sim Poiss(\xi)$
  ☐ `Word Plate :`
  `for all words` $n \in [1, N_m]$ `in document m do`

```
    sample topic index  z_{m,n}  ~  Mult(ϑ_m)
    sample term for word  w_{m,n}  ~  Mult(ψ_{z_{m,n}})
  end for
end for
```

3.4.3 Likelihoods

Probability that a particular word $w_{m,n}$ instantiates a particular term t given the *LDA* parameters is,

$$p(w_{m,n} = t|\vec{\vartheta}_m, \underline{\phi}) = \sum_{k=1}^{K} p(w_{m,n} = t|\vec{\psi}_k)p(z_{m,n} = k|\vec{\vartheta}_m) \tag{3.16}$$

Equation 3.16 corresponds to one iteration on the word plate of the bayesian network.

Joint distribution of all known and hidden variables given the hyperparameters is given by,

$$p(\vec{w}_m, \vec{z}_m, \vec{\vartheta}_m, \underline{\phi}|\vec{\alpha}, \vec{\beta}) = \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\psi}_{z_{m,n}})p(z_{m,n}|\vec{\vartheta}_m).p(\vec{\vartheta}_m|\vec{\alpha}).p(\underline{\phi}|\vec{\beta}) \tag{3.17}$$

Likelihood of one document is given by,

$$p(\vec{w}_m|\vec{\alpha}, \vec{\beta}) = \iint p(\vec{\vartheta}_m|\vec{\alpha}).p(\underline{\phi}|\vec{\beta}). \prod_{n=1}^{N_m} \sum_{z_{m,n}} p(w_{m,n}|\vec{\psi}_{z_{m,n}})p(z_{m,n}|\vec{\vartheta}_m)d\underline{\phi}d\vec{\vartheta}_m \tag{3.18}$$

$$= \iint p(\vec{\vartheta}_m|\vec{\alpha}).p(\underline{\phi}|\vec{\beta}). \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\vartheta}_m, \underline{\phi})d\underline{\phi}d\vec{\vartheta}_m \tag{3.19}$$

Likelihood of the whole corpus $W = \{\vec{w}_m\}_{m=1}^{M}$ is given by,

$$p(W|\vec{\alpha}, \vec{\beta}) = \prod_{m=1}^{M} p(\vec{w}_m|\vec{\alpha}, \vec{\beta}) \tag{3.20}$$

Let us describe the quantities used in the model

M number of documents to generate (const scalar).

K number of topics/mixture components (const scalar).

V number of terms t in vocabulary (const scalar).

$\vec{\alpha}$ hyper-parameter on the mixing proportions ($K - vector$ or scalar if symmetric).

$\vec{\beta}$ hyper-parameter on the mixing components ($K - vector$ or scalar if symmetric).

$\vec{\vartheta}_m$ `parameter notation for p(z|d=m), the topic mixture proportion for document` m.
`One proportion for each document,` $\underline{\theta} = \{\vec{\vartheta}_m\}\ m = 1 \cdots M\ (M \times K\ matrix)$.

$\vec{\psi}_k$ `parameter notation for p(t|z=k), the mixture component of topic` k.
`One component for each topic,` $\underline{\phi} = \{\vec{\psi}_k\}\ k = 1 \cdots K\ (K \times V\ matrix)$.

N_m `document length (document-specific), here modeled with a Poisson distribution with constant parameter` xi.

$z_{m,n}$ `mixture indicator that chooses the topic for the nth word in document m.`

$w_{m,n}$ `term indicator for the nth word in document m.`

3.4.4 Inference via Gibbs Sampling

The exact inference is intractable in case of *LDA*. An approximate inference via *Gibbs* sampling is used.

Gibbs Sampling [Walsh, 2004] is a special case of *Markov-chain Monte Carlo*. *MCMC* can emulate high dimensional probability distributions, $p(\vec{x})$ by the stationary distribution of a *Markov chain*. Each sample is generated for each transition in the chain. This is done after a stationary state of the chain has been reached which happens after a so-called "burn-in period" which eliminates the effect of initialization parameters. In *Gibbs* sampling, the dimensions x_i of the distribution are sampled alternately one at a time, conditioned on the values of all other dimensions, denoted by $\vec{x}_{\neg i}$

Bivariate Case of Gibbs Sampling

Consider a bivariate random variable (x, y), and suppose we wish to compute the marginals, $p(x)$ and $p(y)$. The idea behind the sampler is that it is easier to consider a sequence of distributions, $p(x|y)$ and $p(y|x)$ than obtaining the marginal by integration, $p(x) = \int p(x, y)dy$.

Steps

1. Start with some initial value y_0 for y.
2. Obtain x_0 by generating a random variable from a conditional distribution, $p(x|y = y_0)$.
3. Use x_0 to generate a new value of y_1 drawing from a conditional distribution, $p(y|x = x_0)$.

The sampler proceeds as follows,

1. $x_i \sim p(x|y = y_i)$
2. $y_i \sim p(y|x = x_{i-1})$

Repeating this process k times generates a *Gibbs* sequence of length k, where a subset of points (x_j, y_j) for $1 \leq j \leq m < k$ are taken as simulated draws from the full joint distribution.

Multivariate Case

The value of the k^{th} variable is drawn from the distribution, $p(\theta^{(k)}|\Theta^{\neg k})$ where $\Theta^{\neg k}$ denotes a vector containing all the variables but k.

We draw from the distribution,

$\theta_i^{(k)} \sim p(\theta^{(k)}|\theta^{(1)} = \theta_i^{(1)}, \cdots, \theta^{(k-1)} = \theta_i^{(k-1)}, \theta^{(k+1)} = \theta_{i-1}^{(k+1)}, \cdots, \theta^{(n)} = \theta_{i-1}^{(n)})$

For example, if there are four variables, (w, x, y, z), the sampler becomes

1. $w_i \sim p(w|x = x_{i-1}, y = y_{i-1}, z = z_{i-1})$
2. $x_i \sim p(x|w = w_i, y = y_{i-1}, z = z_{i-1})$
3. $y_i \sim p(y|w = w_i, x = x_i, z = z_{i-1})$
4. $z_i \sim p(z|w = w_i, x = x_i, y = y_i)$

Gibbs Sampling Algorithm

To get a sample $p(x)$,

1. Choose dimension i (random or by permutation)
2. Sample x_i from $p(x_i|\vec{x}_{\neg i})$

$$p(x_i|\vec{x}_{\neg i}) = \frac{p(\vec{x})}{p(\vec{x}_{\neg i})} where, \vec{x} = \{x_i, \vec{x}_{\neg i}\} \tag{3.21}$$

Gibbs Sampling for Models with Hidden Variables

For models containing hidden variable, $\vec{z}$, their posterior given the evidence, $p(\vec{z}|\vec{x})$ is a distribution commonly wanted.

The general formula of a *Gibbs* sampler for such latent variable models becomes,

$$p(z_i|\vec{z}_{\neg i}, \vec{x}) = \frac{p(\vec{z}, \vec{x})}{p(\vec{z}_{\neg i}, \vec{x})} \tag{3.22}$$

LDA Gibbs Sampler

Target of inference is the distribution, $p(\vec{z}|\vec{w})$

$$p(\vec{z}|\vec{w}) = \frac{\vec{z}, \vec{w}}{p(\vec{w})} = \frac{\prod_{i=1}^{W} p(z_i, w_i)}{\prod_{i=1}^{W} \sum_{k=1}^{K} p(z_i = k, w_i)} \tag{3.23}$$

Full conditional,

$$p(z_i|\vec{z}_{\neg i}, \vec{w}) \tag{3.24}$$

is used to simulate $p(\vec{z}|\vec{w})$

This requires the joint distribution,

$$p(\vec{w}, \vec{z}|\vec{\alpha}, \vec{\beta}) = p(\vec{w}|\vec{z}, \vec{\beta})p(\vec{z}|\vec{\alpha}) \tag{3.25}$$

Calculation of $p(\vec{w}|\vec{z}, \vec{\beta})$

W words of the corpus are observed according to independent multinomial trials,

$$p(\vec{w}|\vec{z}, \underline{\phi}) = \prod_{i=1}^{W} p(w_i|z_i) = \prod_{i=1}^{W} \psi_{z_i, w_i} \tag{3.26}$$

Splitting the product over words into product over topics and one over vocabulary,

$$p(\vec{w}|\vec{z}, \underline{\phi}) = \prod_{k=1}^{K} \prod_{t=1}^{V} p(w_i = t|z_i = k) = \prod_{k=1}^{K} \prod_{t=1}^{V} \psi_{k,t}^{n_k^t} \tag{3.27}$$

where, $n_k^{(t)}$ denotes the number of times that the term t has been observed with topic k.

Integrating Equation 3.27 over $\underline{\phi}$ we get,

$$p(\vec{w}|\vec{z}, \vec{\beta}) = \int p(\vec{w}|\vec{z}, \underline{\phi}) p(\underline{\phi}|\vec{\beta}) d\underline{\phi} \tag{3.28}$$

$$= \int \prod_{z=1}^{K} \frac{1}{\Delta(\vec{\beta})} \prod_{t=1}^{V} \psi_{z,t}^{n_z^t + \beta_t - 1} d\vec{\phi}_z \tag{3.29}$$

$$= \prod_{z=1}^{K} \frac{\Delta(\vec{n}_z + \vec{\beta})}{\Delta(\vec{\beta})}, \vec{n}_z = \{n_z^{(t)}\}_{t=1 \cdots N} \tag{3.30}$$

Calculation of $p(\vec{z}|\vec{\alpha})$

$$p(\vec{z}|\underline{\theta}) = \prod_{i=1}^{W} p(z_i|d_i) \tag{3.31}$$

$$= \prod_{m=1}^{M} \prod_{k=1}^{K} p(z_i = k|d_i = m) \tag{3.32}$$

$$= \prod_{m=1}^{M} \prod_{k=1}^{K} \vartheta_{m,k}^{n_m^k} \tag{3.33}$$

where, d_i refers to the document a word i belongs to and $n_m^{(k)}$ refers to the number of times that topic k has been observed with a word of document m.

Integrating Equation 3.33 over $\underline{\theta}$ we get,

$$p(\vec{z}|\vec{\alpha}) = \int p(\vec{z}|\underline{\theta}) p(\underline{\theta}|\vec{\alpha}) d\underline{\theta} \tag{3.34}$$

$$= \int \prod_{m=1}^{M} \frac{1}{\Delta(\vec{\alpha})} \prod_{k=1}^{K} \vartheta_{m,k}^{n_m^k + \alpha_k - 1} d\vec{\vartheta}_m \tag{3.35}$$

$$= \prod_{m=1}^{M} \frac{\Delta(\vec{n}_m + \vec{\alpha})}{\Delta(\vec{\alpha})}, \vec{n}_m = \{n_m^{(k)}\}_{k=1\cdots K} \tag{3.36}$$

Putting Equation 3.36 and Equation 3.30 into Equation 3.25 we get,

$$p(\vec{z}, \vec{w}|\vec{\alpha}, \vec{\beta}) = \prod_{z=1}^{K} \frac{\Delta(\vec{n}_z + \vec{\beta})}{\Delta(\vec{\beta})} \prod_{m=1}^{M} \frac{\Delta(\vec{n}_m + \vec{\alpha})}{\Delta(\vec{\alpha})} \tag{3.37}$$

Equation 3.37 is the joint distribution.

Now, the full conditional given in Equation 3.24 can be expressed as,

$$p(z_i = k|\vec{z}_{\neg i}, \vec{w}) = \frac{p(\vec{w}, \vec{z})}{p(\vec{w}, \vec{z}_{\neg i})} \tag{3.38}$$

$$= \frac{p(\vec{w}|\vec{z})p(\vec{z})}{p(\vec{w}|\vec{z}_{\neg i})p(\vec{z}_{\neg i})} \tag{3.39}$$

$$\propto \frac{\Delta(\vec{n}_z + \vec{\beta})}{\Delta(\vec{\beta})} \cdot \frac{\Delta(\vec{n}_m + \vec{\alpha})}{\Delta(\vec{\alpha})} \tag{3.40}$$

$$\propto \frac{\Gamma(n_k^{(t)} + \beta_t)\Gamma(\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t)}{\Gamma(n_{k,\neg i}^{(t)} + \beta_t)\Gamma(\sum_{t=1}^{V} n_k^{(t)} + \beta_t)} \cdot \frac{\Gamma(n_m^{(k)} + \alpha_k)\Gamma(\sum_{k=1}^{K} n_{m,\neg i}^{(k)} + \beta_t)}{\Gamma(n_{m,\neg i}^{(k)} + \beta_t)\Gamma(\sum_{k=1}^{K} n_m^{(k)} + \beta_t)} \tag{3.41}$$

$$\propto \frac{n_{k,\neg i}^{(t)} + \beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t} \cdot \frac{n_{m,\neg i}^{(k)} + \alpha_k}{[\sum_{k=1}^{K} n_{m,\neg i}^{(k)} + \alpha_k] - 1} \tag{3.42}$$

where the counts $n_{.,\neg i}^{(.)}$ indicate that the token i is excluded from the corresponding document or topic.

Multinomial parameters

The multinomial parameter sets, $\underline{\Theta}$ and $\underline{\phi}$ that correspond to the state of the Markov chain, $M = \vec{w}, \vec{z}$ can be obtained as follows

$$p(\vec{\vartheta}_m|M, \vec{\alpha}) = \frac{1}{Z_{\vartheta_m}} \prod_{n=1}^{N_m} p(z_{m,n}|\vec{\vartheta}_m)p(\vec{\vartheta}_m|\vec{\alpha}) = Dir(\vec{\vartheta}_m|\vec{n}_m + \vec{\alpha}) \tag{3.43}$$

$$p(\vec{\psi}_k|M, \vec{\beta}) = \frac{1}{Z_{\psi_k}} \prod_{k=1}^{K} p(w_i|\vec{\psi}_k)p(\vec{\psi}_k|\vec{\beta}) = Dir(\vec{\psi}_k|\vec{n}_k + \vec{\beta}) \tag{3.44}$$

Using the expectation of the dirichlet distribution, $Dir(\vec{\alpha}) = \frac{a_i}{\sum_i a_i}$ in Equation 3.43 and Equation 3.44 we get,

$$\psi_{k,t} = \frac{n_k^{(t)} + \beta_t}{\sum_{t=1}^{V} n_k^{(t)} + \beta_t} \tag{3.45}$$

$$\vartheta_{m,k} = \frac{n_m^{(k)} + \alpha_t}{\sum_{k=1}^{K} n_m^{(k)} + \alpha_k} \tag{3.46}$$

Gibbs Sampling Algorithm for LDA

☐ Initialization
zero all count variables, $n_m\hat{}(k), n_m, n_k\hat{}(t), n_k$
for all documents $m \in [1, M]$ do
  for all words $n \in [1, N_m]$ in document m do
    sample topic index $z_{m,n} = k \sim Mult(1/K)$
    increment document-topic count: $n_m\hat{}(k) + 1$
    increment document-topic sum: $n_m + 1$
    incremnet topic-term count: $n_k\hat{}(t) + 1$
    increment topic-term sum: $n_k + 1$
  end for
end for
☐ Gibbs sampling over burn-in period and sampling period
while not finished do
  for all documents $m \in [1, M]$ do
    for all words $n \in [1, N_m]$ in docuemnt m do
      ☐ for the current assignment of k to a term t for word $w_{m,n}$:
      decrement counts and sum:$n_m\hat{}k - 1, n_m - 1, n_k\hat{}(t) - 1, n_k - 1$
      ☐ multinomial sampling according to Equation 3.42 (decrements from the previous step)
      sample topic index $\bar{k} \sim p(z_i|\vec{z}_{\neg\ i}, \vec{w})$
      ☐ use the new assignment of $z_{m,n}$ to the term t for word $w_{m,n}$ to:
      increment the counts and sum:$n_m\hat{}k + 1, n_m + 1, n_k\hat{}(t) + 1, n_k + 1$
    end for
  end for
  ☐ check convergence and read out parameters
  if converged ad L sampling iterations since last read out then
    ☐ the different parameters read outs are averaged
    read out parameter set $\underline{\phi}$ according to Equation 3.45
    read out parameter set $\underline{\theta}$ according to Equation 3.46
  end if
end while

3.4.5 Inferencing

Inferencing is the process of finding out the topic distribution in a new document. Suppose the new document is represented by $\bar{m}$, Let us represent a new document by $\vec{w}$. We need to find

out the posterior distribution of topics $\vec{z}$ given the word vector of the document $\vec{w}$ and the *LDA* *Markov* state, $M = \{\vec{z}, \vec{w}\} : p(\vec{z}, \vec{w}; M)$.

The algorithm is a modification the *Gibbs* sampling algorithm we saw. It starts of by randomly assigning topics to words and then performs number of loops through the *Gibbs* sampling update (locally for the words i of $\bar{m}$) [Heinrich, 2005].

$$p(\bar{z}_i = k | \bar{w}_i = t, \bar{\vec{z}}_{\neg i}, \bar{\vec{w}}_{\neg i}; M) = \frac{n_k^{(t)} + \bar{n}_{k,\neg i}^{(t)} + \beta_t}{\sum_{t=1}^{V} n_k^{(t)} + \bar{n}_{k,\neg i}^{(t)} + \beta_t} \cdot \frac{n_{\bar{m},\neg i}^{(k)} + \alpha_k}{[\sum_{k=1}^{K} n_{\bar{m},\neg i}^{(k)} + \alpha_k] - 1} \tag{3.47}$$

where $\bar{n}_k^t$ counts the observations of term t and topic k in the new document.

The topic distribution of the new document can be found out using the following equation,

$$\vartheta_{\bar{m},k} = \frac{n_{\bar{m}}^{(k)} + \alpha_k}{\sum_{k=1}^{K} n_{\bar{m}}^{(k)} + \alpha_k} \tag{3.48}$$

In the next section, we will evaluate *LDA*.

3.5 Evaluation of LDA

Evaluation of topic models has been discussed at length in [Wallach, Murray, Salakhutdinov, and Mimno, 2009]. A natural evaluation metric discussed in [Wallach, Murray, Salakhutdinov, and Mimno, 2009] is finding out the probability of a held-out document given a trained model. Topic modeling is a useful tool for analyzing unstructured text collections. Evaluation of topic models is difficult due to their unsupervised nature. For some applications, there might be extrinsic tasks such as information retrieval for which performance can be evaluated. There is a need for a universal method that measures generalization capability of a topic model in a way that is accurate, computationally efficient and independent of a specific application. *LDA* can be evaluated by 1) Information retrieval accuracy or 2) by estimating the probability of unseen held-out documents given some training documents. We propose a new evaluation method as follows.

Evaluation method

1. Download documents.
2. Tag every document with a topic to get a tagged corpus
3. Held-out some documents for testing before training the model

4. Train the topic model usingd the untagged corpus (obtained after removing the tags)
5. Now, use the trained model to infer topic distribution for the testing documents
6. Check whether the topic having highest proportion matches the tag of the document

5-fold cross validation was used in this case. 1273 documents were downloaded from DMOZ [dmoz open directory project, 2013]. Computers, films, real estate, cooking and sports were the 5 topics chosen. The implementation in Mallet was used to conduct the experiment.

Topic	No. of files
Computers	164
Sports	213
Cooking	251
Real Estate	261
Films	384

Table 3.1 Number of files per topic

Average accuracy	20.867

Table 3.2 Average Accuracy

Discussion

The average accuracy after 5-fold cross-validation on this corpus was 20.867, which is very low. The reason for this is the short-length of the documents used. *LDA* works on the principle of co-occurrence. If we look at Equation 3.42, there is a factor for words and another for documents. Probabilities are higher for assignments that "don't break document boundaries", that is, words appearing in the same document have a slightly higher odds of ending up in the same topic. The same holds for document assignments, they to a degree follow "word boundaries". These effects mix up and spread over clusters of documents and words, eventually. Due to the short length of the documents, words from the same topic may not always co-occur. Also, there is chance of them co-occurring with words from other topics also which results in bad clustering. Some words belong to more than one topic due to this. Due to this, the clustering of documents as whole in this case is not good.

Also, the evaluation method used is very strict. If it is a bit lenient, the accuracy can be increased. A new method called weighted evaluation can be used in this case.

Weighted Evaluation

Weighted evaluation is based on the fact that we get a topic distribution for each testing document. This topic distribution is arranged in descending order of topic proportions. The idea is to assign weights according to the rank given to the original tag of the document.

Weighted Evaluation Algorithm

```
matches=0, counts=0
For each document
  Find topic distribution
  Switch(tag):
    case(topic1): matches += 1
    case(topic2): matches += 0.8
    case(topic3): matches += 0.6
    case(topic4): matches += 0.4
    case(topic5): matches += 0.2
    counts++
Accuracy = matches/counts
```

Results

Fold	Matches	Counts	Accuracy (in percentage)
Fold 1	156	255	61.4
Fold 2	156	255	61.4
Fold 3	170	255	66.9
Fold 4	152	255	59.6
Fold 5	161	253	63.9

Table 3.3 Accuracy for each testing fold

Average accuracy	62.6

Table 3.4 Weighted Evaluation Average Accuracy

Discussion

As we can see the accuracy has increased after we used weighted evaluation. This kind of evaluation needs to be used in many systems including transliteration where the most probable

word needs to be predicted. The rank of the actual word may be further down. This doesn't mean that the system is giving wrong output. Therefore, a more lenient approach would be better in this case.

The accuracy has increased but still is unsatisfactory. 62 % accuracy in this case implies that given a document, there is 62 % chance that the main topic of the document will be ranked in top 5. As we have used only 5 topics, this result is not that significant. Using more number of topics will lead to better insights.

If we consider the clustering to be effective only till the third rank, we get accuracies as shown in the table.

Fold	Matches	Counts	Accuracy (in percentage)
Fold 1	156	255	50.9
Fold 2	156	255	50.1
Fold 3	170	255	57.4
Fold 4	152	255	46.7
Fold 5	161	253	53.5

Table 3.5 Accuracy for each testing fold (Till 3rd rank)

Average accuracy	51.7

Table 3.6 Average Accuracy (Till 3rd rank)

This means that given a document, there is 51.7 % chance that the main topic of the document will be ranked in top 3. It is known that *LDA* performs better with more number of topics. So, increasing the number of topics while performing the evaluation can lead to more better results.

A list of high probability words for each topic was prepared which is given in the following table.

Computer	Films	Cooking	Real Estate	Sports
site	film	recipes	services	reviews
software	information	recipe	real	news
free	offers	including	company	interviews
systems	production	collection	estate	information
programming	courses	tips	includes	features
research	links	source	commercial	current
resources	videos	production	based	tennis
code	television	baking	development	running
information	cinema	breakfast	title	tournament

Table 3.7 High probability words in each topic

As we can see, the high probability words in each topic are good but due to the short length, results were not that good.

SUMMARY

We started with an introduction of *IR*. Popular approaches for IR were listed. We had a detailed discussion on corpus models like multivariate binary model, poisson model, multinomial model, dirichlet smoothed model, *DCM* and *LDA*. The generative modeling it uses, inference procedure and evaluation.

The main goal of this project is to make use of generative models for sentiment analysis. In the next chapter we focus on some existing work in this direction. We also show how can *LDA* and topical n-grams model be used for sentiment analysis.

Chapter 4

Sentiment Analysis using Topic models

In this chapter, we will show how topic models are used for *SA*. In the first section, we go through some of the related work where sentiment and topic have been jointly modeled. After that, we show how the basic *LDA* model can be used to classify documents based on their polarity. This is followed by the explanation of the *JST* model in detail. The use of *Topical n-grams* model for *SA* is shown in the section following this. The experimental setup, results, and error analysis is done in Section 4.5.

Joint sentiment and topic models have been used to tackle the sentiment classification problem. Despite having a hierarchical structure, these generative models have a bag of words assumption. Due to this fact, they tend to misclassify texts having sentiment in the form of phrases. *LDA* and it's extensions don't work properly with phrases. To tackle this situation, we propose an unsupervised approach to sentiment analysis using the topical n-grams model which has been shown to be effective with phrases. We train the topical n-grams model using two topics i.e., positive, and negative, list of positive and negative words, and rules to detect positive and negative phrases. New documents are then classified using this trained model. The system gives better results than the existing Joint Sentiment Topic model. We also propose an approach to generate a list of positive and negative words using LDA based on our observations reported in Section 4.5.

In the next section, we will point out some of the related work in this direction.

4.1 Related Work

There are many supervised, semi-supervised and unsupervised approaches to solve the sentiment analysis problem. One supervised method based on n-gram analysis is explained by [Bespalov, Bai, Qi, and Shokoufandeh, 2011]. In this they map the n-grams to a low-dimensional latent semantic space where a classification function can be defined. Our approach being unsupervised, we will go through some of the related work in that direction. One more motivation to consider only semi-supervised and unsupervised approaches is the fact that they don't need any corpus and don't have any domain specific limitations. Rule based systems can also be considered as unsupervised systems but usually due to exceptions to these rules, their performance is affected. Due to this, we are not considering them in further discussion.

[Turney, 2002] used an unsupervised learning algorithm based on mutual information between document phrases and a small set of positive/negative paradigm words called seed words to classify the semantic orientation at the word/phrase level. Another unsupervised approach to classify the text at document level was proposed by [Turney and Littman, 2002]. [Eguchi and Lavrenko, 2006] created a generative model that jointly models sentiment words, topic words and sentiment polarity in a sentence as a triple. [Mei, Ling, Wondra, Su, and Zhai, 2007] proposed another generative model, called TSM (Topic Sentiment Mixture) model which can be used to discover topics in blogs as well as their associated sentiments. A novel generation model that unifies topic-relevance and opinion generation by a quadratic combination was proposed by [Zhang and Ye, 2008]. A probabilistic generative model based on LDA called JST (Joint Sentiment Topic) model was shown to perform well for sentiment analysis of reviews by [Lin and He, 2009]. It is a fully unsupervised method and shows good result when priors are used for training. Another extension of LDA which tries to unify aspect and sentiment was proposed by [Jo and Oh, 2011]. All the unsupervised methods using generative models discussed here operate at the word level. Due to this they lose out on the information provided by phrases which may lead to incorrect classification.

In the next section, we will show how to use the basic *LDA* model for sentiment analysis.

4.2 LDA for sentiment analysis

Let us first list the basic steps to use any topic model for discovering topics.

4.2.1 Using Topic models

- Set number of topics.

- Remove stop-words as they do not belong to any topic.
- Estimate probabilities using some inference method.
- Use the trained model for inference of topics in new documents.

4.2.2 Using LDA for Sentiment Classification

To use basic LDA as a sentiment classifier, we add one more step to remove objective words. Also, usually during Gibbs sampling the first step assigns topics randomly to words. Instead, we introduce a prior information about the positivity and negativity of words to assign topics to words initially. The steps are as follows.

- Set number of topics, 2 in this case viz. positive and negative.
- Remove stop-words.
- Remove objective words as they won't affect sentiment.
- **Gibbs Sampling** with prior using lists of positive and negative words. Gibbs Sampling was explained in detail in Chapter 3. We will again explain it briefly here.
  - Go through each document, and randomly assign each word in the document to one of the K topics. this random assignment already gives you both topic representations of all the documents and word distributions of all the topics (albeit not very good ones).
  - So to improve on them, for each document d, go through each word w in d, and for each topic t, compute two things.
    1. $p(t|d)$ = the proportion of words in document d that are currently assigned to topic t.
    2. $p(w|t)$ = the proportion of assignments to topic t over all documents that come from this word w.
  - After this, Reassign w a new topic, where you choose topic t with probability $p(t|d) \times p(w|t)$. According to the generative model, this is essentially the probability that topic t generated word w, so it makes sense that we re-sample the current word's topic with this probability. In this step, we're assuming that all topic assignments except for the current word in question are correct, and then updating the assignment of the current word using our model of how documents are generated.
  - After a suitable number of iterations, we get a proper probability distribution.

As explained here, the **first step assigns topics randomly**. In our case, we make **use of prior knowledge which a list of positive and negative words to assign the positive or negative topic to each word** initially. The inclusion of prior increases the accuracy of the system as shown in Section 4.5.

- Use the trained model to classify a new document as positive or negative.

Having prior knowledge in the case of LDA, means we have the word-topic distribution as shown in Figure 4.1.

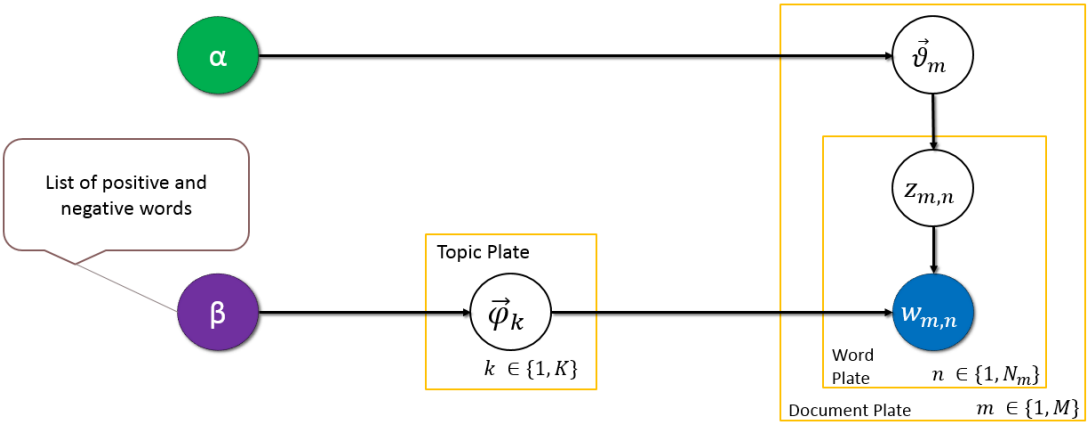


Figure 4.1 LDA with prior

4.2.3 Example

Let us consider an example to see how it works.

Text Input

Okay, so Janet is an AMAZING actress, I love her brother MJ so thats why I rented this. I was amazed at the emotion she put into her character, this movie is NOT for young children! She did amazing channeling the anger that Patricia had to potray, I think this is a MUST SEE movie!

After converting to lower case

okay, so janet is an amazing actress, i love her brother mj so thats why i rented this. i was amazed at the emotion she put into her character, this movie is not for young children ! she did amazing channeling the anger that patricia had to potray, i think this is a must see movie!

After removing stop-words

okay, so janet is an amazing actress, i love her brother mj so thats why i rented this. i was amazed at the emotion she put into her character, this movie is not for young children ! she did amazing channeling the anger that patricia had to potray, i think this is a must see movie!

After removing objective words

okay, so janet is an amazing actress, i love her brother mj so thats why i rented this. i was amazed at the emotion she put into her character, this movie is not for young children ! she did amazing channeling the anger that patricia had to potray, i think this is a must see movie!

After assigning topic to each subjective word using the word lists

okay, so janet is an amazing actress, i love her brother mj so thats why i rented this. i was amazed at the emotion she put into her character, this movie is not for young children ! she did amazing channeling the anger that patricia had to potray, i think this is a must see movie!

After this the Gibbs sampling procedure is performed on the input text. The final outcome depends on the counts of positive and negative words in the document. But as the sampling procedure ensures that co-occuring words belong to the same topic, the system performs well in many cases as can be seen in the experimental results shown in Section 4.5. This is similar to the bag of words model where the feature vector contains only subjective words.

In the next two sections, we will go through the joint models of topic and sentiment.

4.3 Joint models of topic and sentiment

SA has been used in IR to improve the performance. IR was mainly concerned with factual/objective data. So, intuitively we see that subjectivity classification can aid IR. [Riloff, Wiebe, and Phillips, 2005] has work based on it in which they try to exploit subjectivity analysis to improve performance of information extraction.

Corpus models are useful in fetching documents specific to a certain topic. Sometimes a user might need to fetch documents which have a specific sentiment. One such work on sentiment retrieval using generative models is seen in [Eguchi and Lavrenko, 2006]. In this work, they have assumed that user inputs both query terms as well as indicates the desired sentiment polarity in some way. They have combined sentiment and topic relevance models to retrieve documents which are most relevant to such user requests. This approach is very important for sentiment aware information retrieval.

The expression of sentiment in the text is topic dependent. Negative review for a voting event may be expressed using *flawed*. On the other hand negative review of politician may be expressed using *reckless*. Sentiment polarity is topic dependent [Engström, 2004]. The adjective *unpredictable*

will have a negative orientation in a car review and it will have a positive orientation in a movie review.

4.3.1 Terminology

The goal of the model is to generate a collection of sentences $s_1, s_2, \ldots, s_n$. Every document is composed of words $w_1, w_2, \ldots, w_n$ drawn from the vocabulary V. A binary variable $b_{ij} \in \{S, T\}$ is used to represent whether a word in position j in sentence i is a topic word or a sentiment word. Let x_i be the polarity for the sentence s_i. x_i is a discrete random variable with three outcomes $\{-1, 0, +1\}$. A statement s_i is represented as a set $\{w_i^s, w_i^t, x_i\}$ where w_i^s are the sentiment bearing words, w_i^t are the topic bearing words and x_i is the sentence polarity. The user query will be represented in a similar fashion $\{q_i^s, q_i^t, q^x\}$. Let p denote a unigram language model. P denotes the set of all possible language models. It is the probability simplex. Similarly, let p_x denote the distribution over three possible polarity values and P_x will be the corresponding ternary probability simplex. The function $\pi : P \times P \times P_x \rightarrow [0, 1]$ is a function which assigns a probability $\pi(p_1, p_2, p_x)$ to a pair of language models p_1 and p_2 together with p_x.

4.3.2 Generative model of sentiment

A sentence s_i containing words $w_1, w_2, \ldots, w_j, \ldots, w_m$ is generated in the following way:

1. Draw p_t, p_s*and* p_x from $\pi(\cdot, \cdot, \cdot)$.
2. Sample x_i from a polarity distribution $p_x(\cdot)$.
3. For each position $j = 1 \ldots m$:
   - if $b_{ij} = T$: draw w_j from $p_t(\cdot)$;
   - if $b_{ij} = S$: draw w_j from $p_s(\cdot)$

The probability of observing the new statement s_i containing words $w_1, w_2, \ldots, w_j, \ldots, w_m$ is given by:

$$\sum_{p_t, p_s, p_x} \pi(p_t, p_s, p_x) p_x(x_i) \prod_{j=1}^{m} \begin{cases} p_t(w_j) & \text{if } \mathrm{b_{ij}} = \mathrm{T} \\ p_s(w_j) & \text{otherwise} \end{cases} \tag{4.1}$$

The probability functions are dirichlet smoothed models and $\pi(p_1, p_2, p_x)$ is a non-parametric function.

Each sentence is represented as a bag of words model and the model makes strong independence assumptions. But, due to joint probability distribution used it is able to model co-occurrence.

Retrieval using the model

Suppose we are given a collection of statement C and a query $\{q_i^s, q_i^t, q^x\}$ given by the user. The topic relevance model R_t and the sentiment relevance model R_t are estimated. For each word w in a statement within a collection C, these models are estimated as follows:

$$R_t(w) = \frac{P(q^s, q^t \circ w, q^x)}{P(q^s, q^t, q^x)}, R_s(w) = \frac{P(q^s \circ w, q^t, q^x)}{P(q^s, q^t, q^x)} \tag{4.2}$$

$q \circ w$ means appending w to the list q. The statements are ranked using a variation of cross-entropy,

$$\alpha \sum_v R_t(v) \log p_t(v) + (1-\alpha) \sum_v R_s(v) \log p_s(v) \tag{4.3}$$

The experiments using this approach have shown promising results. This shows that sentiment aware IR can benefit from this technique. As corpus models have been widely used in IR, extending and tuning them for SA aware IR can yield good results.

4.3.3 Joint Sentiment-Topic modeling (JST)

[Lin and He, 2009] discusses a joint model of sentiment and topics. Following figure shows the model.

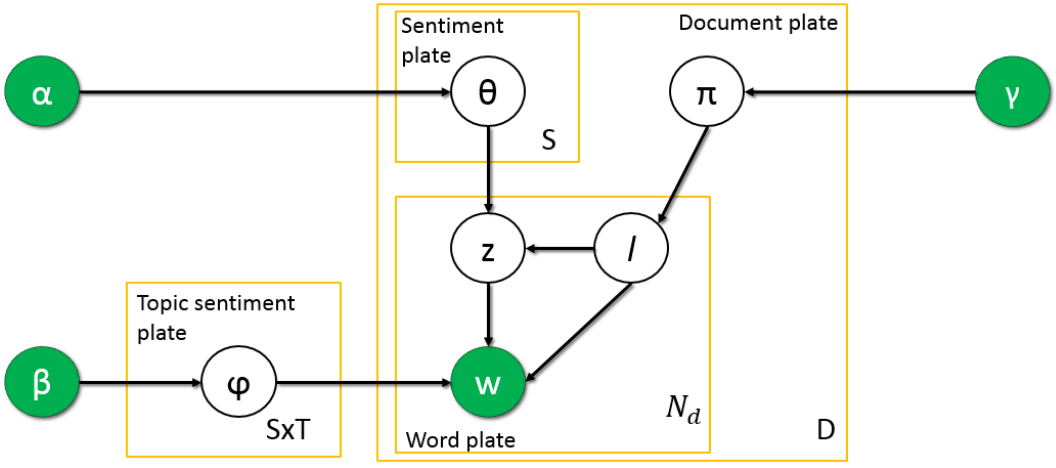


Figure 4.2 Joint Sentiment Topic Model

Assume that we have a collection of D documents denoted by $C = d_1, d_2, \cdots, d_D$; each document in the corpus is a sequence of N_d words denoted by $d = (w_1, w_2, \cdots, w_{N_d})$ and each word in the document is an item from a vocabulary index with V distinct terms denoted by $1, 2, \cdots, V$. Let

S and T be the number of distinct sentiment and topic labels respectively. The procedure of generating a document is described as follows.

For each document d, choose a distribution $\pi_d \sim Dir(\gamma)$.
For each sentiment label l under document d, choose a distribution
$\theta_{d,k} \sim Dir(\alpha)$.
For each word w_i in document d
- choose a sentiment label $l_i \sim \pi_d$,
- choose a topic $z_i \sim \theta_{d,l_i}$,
- choose a word w_i from the distribution over words defined by the topic z_i and sentiment label l_i, $\psi_{z_i}\hat{}l_i$

The hyper-parameter α in JST is the prior observation count for the number of times topic j is associated with sentiment label l sampled from a document.

The hyper-parameter β is the prior observation count for the number of times words sampled from topic j are associated with sentiment label l.

Similarly, the hyper-parameter γ is the prior observation count for the number of times sentiment label l is associated with a document.

The latent variables of interest in JST are

1. The joint sentiment/topic-document distribution, θ
2. The joint sentiment/topic-word distribution, ϕ
3. The joint sentiment-document distribution, π

To obtain the distributions for θ, ϕ, and π, we firstly estimate the posterior distribution over z i.e, the assignment of word tokens to topics and sentiment labels.

We need to estimate the distribution, $P(z_t = j, l_t = k|w, z_{\neg t}, l_{\neg t}, \alpha, \beta, \gamma)$ where $z_{\neg t}$ and $l_{\neg t}$ are vector of assignments of topics and labels for all words in the collection except for the word position t in document d.

The joint distribution can be given as follows,

$$P(w, z, l) = P(w|z, l)P(z|l) = P(w|z, l)P(z|l, d)P(l|d) \tag{4.4}$$

After calculations similar to LDA, we get the following full conditional,

$$P(z_t = j, l_t = k|w, z_{\neg t}, l_{\neg t}, \alpha, \beta, \gamma) = \frac{\{N_{i,j,k}\}_{\neg t} + \beta}{\{N_{j,k}\}_{\neg t} + V\beta} \cdot \frac{\{N_{j,k,d}\}_{\neg t} + \alpha}{\{N_{k,d}\}_{\neg t} + T\alpha} \cdot \frac{\{N_{k,d}\}_{\neg t} + \gamma}{\{N_d\}_{\neg t} + S\gamma} \tag{4.5}$$

where,

V is the size of the vocabulary

T is the number of topics

S is the total number of sentiment labels

D is the number of documents in the collection

$N_{i,j,k}$ is the number of times word i appeared in topic j and with sentiment label k

$N_{j,k}$ is the number of times words are assigned to topic j and sentiment label k

$N_{j,k,d}$ is the number of times a word from document d has been associated with topic j and sentiment label k

$N_{k,d}$ is the number of times sentiment label k has been assigned to some word tokens in document d

N_d is the total number of words in the collection

θ, ϕ, and π can be estimated as follows

$$\phi_{i,j,k} = \frac{N_{i,j,k} + \beta}{N_{j,k} + V\beta} \tag{4.6}$$

$$\theta_{j,k,d} = \frac{N_{j,k,d} + \alpha}{N_{k,d} + T\alpha} \tag{4.7}$$

$$\pi_{k,d} = \frac{N_{k,d} + \gamma}{N_d + S\gamma} \tag{4.8}$$

The Gibbs sampling procedure in this case is similar to that of *LDA*.

JST can be used for document level sentiment classification and topic detection simultaneously. *Joint sentiment topic modeling* is completely unsupervised as compared to existing approaches for sentiment classification. The performance of *JST* on movie review classification is competitive compared to other supervised approaches. *JST* has been used in our experiments to compare it against our approaches for sentiment analysis as explained in Section 4.5.

LDA (Latent Dirichlet Allocation) as shown by [Blei, Ng, and Jordan, 2003] is a generative model used to discover topics in a document collection. It gives two types of distributions as output, document-topic and word-topic distributions. The document-topic distribution gives the

proportion of topics in each document and the word-topic distribution gives the probability of a word being in each topic. LDA works on the principle of co-occurrence. It assumes that words tending to appear together belong to the same topic. JST (Joint Sentiment Topic) as explained by [Lin and He, 2009] is a probabilistic generative model which extends LDA and discovers both sentiment and topic simultaneously in a document collection. JST has shown promising results on binary sentiment classification.

There are many extensions of the basic LDA model which try to combine both sentiment and topic to solve the problem of sentiment analysis. All these models including LDA have one underlying assumption which makes them unsuitable for text classification purposes. They assume that each word is generated separately and independent of other words. This is essentially the bag of words assumption. However, text being a sequence of words, the correct meaning of the text cannot be understood by merely capturing co-occurrences. In addition to this, we also need to consider collocation of words. A phrase is a collocation of words which usually has more meaning than the individual words making up that phrase. There is a subtle difference between a phrase and collocation of words. Not all collocations of words can be considered as a phrase. We need a model which takes into account phrases to completely understand the meaning of the text.

Topical n-grams model proposed by [Wang, McCallum, and Wei, 2007] is one such generative model which takes into account not only co-occurrences but also collocations of words. It also decides whether a particular collocation of words should be considered as a phrase or not. We train this model using 3 topics viz. positive, negative and objective, using a prior list of positive and negative words, and some rules to identify the subjective nature of phrases. The phrases in our experiments are restricted to bigrams. We then use this trained model to infer the topic distribution for new documents. The topic having higher proportion is considered to be the class of the document.

In the next section, we will explain the topical n-grams model.

4.4 Topical n-grams model

n-gram phrases (or collocations) are fundamentally important in many areas of natural language processing (e.g., parsing, machine translation and information retrieval). Phrase as the whole carries more information than the sum of its individual components, thus it is much more crucial in determining the topics of document collections than individual words [Wang and McCallum, 2005]. However, most of the topic models assume that words are generated independently to each other, i.e., under the bag of words assumption. The possible over complicacy caused by introducing phrases makes these topic models completely ignore them. It is true that these

models with the bag of words assumption have enjoyed a big success, and attracted a lot of interests from researchers with different backgrounds. A topic model considering phrases would be more useful in certain applications. Topical n-grams model is one such generative model. It's generative process is explained as follows,

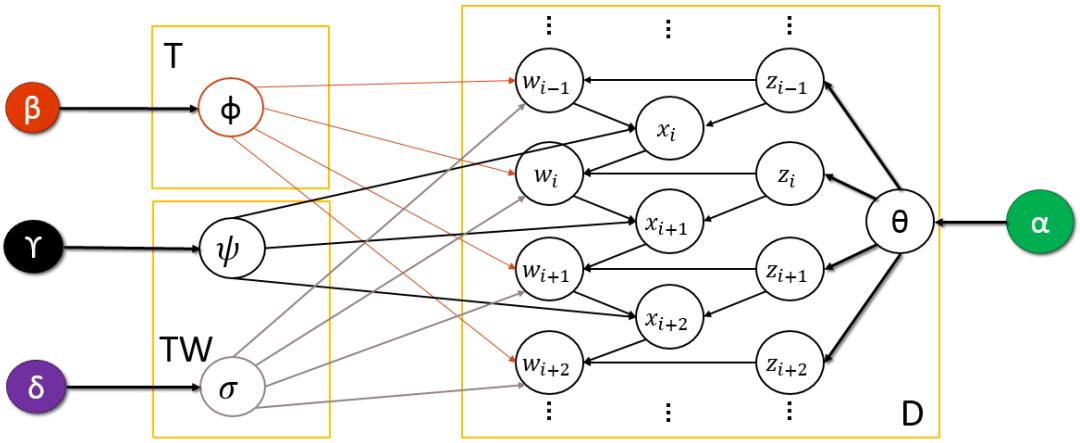


Figure 4.3 Topical n-grams model

1. Draw multinomial ϕ_z from a Dirichlet prior β;
2. Draw binomial ψ_z from a Beta prior γ;
3. Draw multinomial σ_{zw} from a Dirichlet prior δ;
4. For each document d, draw a multinomial ${}^{(d)}$ from a Dirichlet prior α; then for each word ${w_i}^{(d)}$ in document d,
   (a) Draw ${x_i}^{(d)}$ from binomial $\psi_{{w_{i-1}}^{(d)}}$;
   (b) Draw ${z_i}^{(d)}$ from multinomial $\theta_{(d)}$;
   (c) Draw ${w_i}^{(d)}$ from multinomial $\sigma_{{w_{i-1}}^{(d)}}$ if ${x_i}^{(d)} = 1$; else draw ${w_i}^{d}$ from multinomial $\phi_{{z_i}^{(d)}}$.

The main point to infer from this generative process is that the topic assignments for the two terms in a bigram are not required to be identical. In the description of topic n-grams given in [Wang and McCallum, 2005], they have used the topic of the last term as the topic of the phrase. But in our experiments, we have used certain rules as prior information to assign topics to a phrase initially.

4.4.1 Inferencing

Gibbs sampling is used to conduct approximate inference in this paper. During Gibbs sampling, we draw the topic assignment z_i and the bigram status x_i iteratively for each word w_i according to the following conditional probability distribution:

$$p(z_i, x_i | z_{-i}, w_{-i}, w, \alpha, \beta, \gamma, \delta) \propto \frac{\gamma_{x_i} + p_{z_{i-1} w_{i-1} x_i}}{\sum_{k=0}^{1} (\gamma_k + p_{z_{i-1}} w_{i-1} k)} (\alpha_{z_i} + q_{d_{z_i}}) \times \begin{cases} \frac{\beta_{w_i} + n_{z_i w_i}}{\sum_{v=1}^{V} (\beta_v + n_{z_i v})} & \text{if } x_i \text{ is even} \\ \frac{\delta_{w_i} + m_{z_i w_{i-1} w_i}}{\sum_{v=1}^{V} (\delta_v + m_{z_i w_{i-1} v})} & \text{if } n \text{ is odd} \end{cases} \tag{4.9}$$

where,

z_{-i} denotes the topic assignments for all word tokens except word w_i,
x_{-i} represents the bigram status for all tokens except word w_i,
n_{zw} represents how many times word w is assigned into topic z as a unigram,
m_{zwv} represents how many tmes word v is assigned as the second term of a bigram given the previous word w,
p_{zwk} denotes how many times the status variable x equals k given the previous word w and previous word's topic z, and
q_{dz} represents how many times a word is assigned to topic z in document d.

In next section, we will explain the use of topical n-grams model for sentiment analysis

4.4.2 Topical n-grams model for Sentiment Analysis

To make use of topical n-grams model for sentiment classification, we use an approach similar to using LDA for sentiment classification.

- Set number of topics equal to 2.
- Remove stop-words.
- Remove objective words as they won't affect sentiment. The objective words in this case do not include the negation words like *doesn't, won't, no*, etc. This is to ensure that we can catch negation of polarity when they are used with subjective words.
- Apply Gibbs Sampling with prior. The prior used in this case is more sophisticated and can handle both words and phrases. In case of words, if is not present in a bigram then simply use a list of positive and negative words to assign positive or negative topic. If the word is present in the next bigram then assign it the topic of the bigram. There are some rules to detect and assign topics to bigrams which are explained next.

- Use the trained model to classify a new document as positive or negative.

Rules for Topic assignment of phrases

At present, our rules are restricted to bigrams. We plan to extend them as explained in Section Chapter 6. In the following rules, we mean topic when we say polarity. The use of polarity makes it easy to understand the rules as they are concerned with subjectivity.

1. If the first word in the bigram is a negation word and the second word is subjective then the polarity of the bigram is opposite to the polarity of the second word.
   Examples: *won't like, won't regret, etc.*. Here, *won't like* is assigned negative polarity and *won't regret* is assigned positive polarity.

2. If both the words in the bigram are subjective then are two cases. If both words are of the same polarity then resultant polarity is the same. But if their polarities are different, then the polarity of the first word is assigned to the bigram.
   Examples: *beautifully amazing* is positive as both words as positive. *lack respect* is assigned negative as per the rules.

One salient feature of this approach is that it is an unsupervised method and can work on any domain.

Let us explain the procedure using an example

4.4.3 Example

Text Input

I was a little worried that Tyler wouldn't be able to pull off a sequel but he did it. Even though it wasn't as good as the first it was funnier in my opinion. The storyline was decent carrying over the drama from the first and the ending to me seemed like a quick way to rap it up. I do not regret buying this movie and I would not mind watching it again. I saw this one as more of a comedy even though it had it's sad parts. I would recommend it if you enjoyed the first.

Converting to lowercase

i was a little worried that tyler wouldn't be able to pull off a sequel but he did it. even though it wasn't as good as the first it was funnier in my opinion. the storyline was decent carrying over the drama from the first and the ending to me seemed like a quick way to rap it up. i do not regret buying this movie and i would not mind watching it again. i saw this one as more of a comedy even though it had it's sad parts. i would recommend it if you enjoyed the first.

After removing stop-words except negation words

i was a little worried that tyler wouldn't be able to pull off a sequel but he did it. even though it wasn't as good as the first it was funnier in my opinion. the storyline was decent carrying over the drama from the first and the ending to me seemed like a quick way to rap it up. i do not regret buying this movie and i would not mind watching it again. i saw this one as more of a comedy even though it had it's sad parts. i would recommend it if you enjoyed the first.

After removing objective words

i was a little worried that tyler wouldn't be able to pull off a sequel but he did it. even though it wasn't as good as the first it was funnier in my opinion. the storyline was decent carrying over the drama from the first and the ending to me seemed like a quick way to rap it up. i do not regret buying this movie and i would not mind watching it again. i saw this one as more of a comedy even though it had it's sad parts. i would recommend it if you enjoyed the first.

The input text to form bigrams

worried wouldn't. wasn't good funnier. decent. not regret. not mind. sad. recommend enjoyed

After forming bigrams

worried wouldn't worried_wouldn't. wasn't good wasn't_good funnier good_funnier. decent. not regret not_regret. not mind not_mind. sad. recommend enjoyed recommend_enjoyed.

After assigning topics to words and bigrams

worried wouldn't worried_wouldn't. wasn't good wasn't_good funnier good_funnier. decent. not regret not_regret. not mind not_mind. sad. recommend enjoyed recommend_enjoyed.

here, the blue color means that the topic is assigned randomly, red color indicated negative polarity, and green color indicates positive polarity. We can see that most of the grams here are positive. Therefore, at the end, the document is assigned a positive polarity which is the correct classification.

4.5 Experiments

Analysis was performed for the binary sentiment classification task. The language used in this case was English. We conducted experiments on 4 models, BOW using SVM, LDA [Blei, Ng, and Jordan, 2003], JST [Lin and He, 2009], and Topical n-gram model [Wang, McCallum, and Wei, 2007]. We used two settings for the topic models, with and without prior. The word lists used are those specified in [Liu, 2010]. The implementations of SVM, LDA and Topical

n-gram in Mallet [1] have been used for evaluation. For JST, we have used the implementation provided by the authors [2]. The default settings for the hyper-parameters were used in all these implementations. Some changes in the implementations of LDA and Topical n-gram were made to take into account the prior information.

4.5.1 Dataset

To create the dataset we used the amazon reviews dataset provided by SNAP [3]. These reviews are not tagged with sentiment but they have ratings from 1 to 5. We used this information to create a sentiment tagged corpus. The reviews with ratings less than 3 were tagged as negative and others were tagged as positive. We conducted the experiments on 6,00,000 reviews containing equal number of positive and negative reviews.

4.5.2 Results

The results presented here are for 10-fold cross validation. For the topic models, due to the randomness involved during sampling, the best result obtained for each fold has been used for calculating the average. The Bag of words system performs better than all the models when no prior information is provided. The performance of topic models significantly increases when prior information is provided. Our approach to use Topical n-gram outperforms all the systems when a prior is used.

System	**Avg. accuracy (%)**
BOW-SVM	82.45
LDA	65.34
LDA with prior	80.19
JST	68.64
JST with prior	84.43
Topical n-gram	63.57
Topical n-gram with prior	**87.32**

Table 4.1 Evaluation of Topic models for Binary sentiment classification.

4.5.3 Discussion

The performance of our system is better due to the capacity to handle phrases. All the other systems, consider each word separately. The performance improvement over JST is 3% which

[1] http://mallet.cs.umass.edu/
[2] https://github.com/linron84/JST
[3] http://snap.stanford.edu/

is statistically significant. Though the system performs better for this dataset, it still has some obvious limitations. The system highly depends on the rules used for initial assignment of topics which is evident from the results. These rules don't apply to all the bigrams. Let us consider, the bigram *insanely good*. In this case, the first word is negative and second word is positive. The rules will assign a negative polarity to this bigram. But in this phrase *insanely* is used to increase the intensity of *good*. The rules apply only for bigrams, we need to add more rules to handle n-grams. A phrase like *I don't think it's good* won't be handled by the system. During error analysis, we also found that some words like *engrossing, blockbuster, bravo, etc.*, are not present in the word lists. These words convey a positive sentiment but our system fails to correctly classify reviews containing such words. Also, words like *awsome* which is spelling mistake of *awesome* are often found in reviews. We thought that the accuracy might be increased if we could detect the correct subjective nature of these words and also the bigrams using them. For this, we proposed an approach for resource generation using LDA.

In the next section, we explain an approach to do the same.

4.6 Resource generation using LDA

LDA can be used for resource generation of positive and negative words. The steps to do so are explained below.

- Set number of topics equal to 3 i.e., positive, negative and objective. We do not remove the objective words in this case as we want to find out which of them are positive or negative.
- Remove stop-words.
- Gibbs Sampling with prior using lists of positive and negative words. In the initial step, words present in the list are assigned that specific topic but the other words as assigned a topic randomly.
- Get the top words in the positive and negative topics.

The list of positive and negative words we obtained using this approach were used to enhance our systems as show in Section 4.5.

The word lists we generated using this approach were used to test the systems using priors. The results for the same are shown in Table 2. We can see a marginal increase in the accuracy in this setting.

System	**Avg. accuracy (%)**
LDA with prior	80.21
JST with prior	86.37
Topical n-gram with prior	**89.83**

Table 4.2 Evaluation using resources generated using *LDA*

Summary

In this chapter we explained how basic LDA can be used to perform the sentiment classification task. We also studied two models which combine sentiment and topics. Both the systems have shown high competence in classification tasks. We also discussed the the topical n-gram model in detail and explained how it can be used for sentiment classification. After that, we showed through experiments that this technique actually work and perform better than the bag of words and *JST* model.

In the next chapter, we will see how deep semantics can be used for sentiment analysis.

Chapter 5

Deep Semantics for Sentiment Analysis

Existing methods for sentiment analysis use supervised approaches which take into account all the subjective words and or phrases. Due to this, the fact that not all of these words and phrases actually contribute to the overall sentiment of the *text* is ignored. We propose an unsupervised rule-based approach using deep semantic processing to identify only relevant subjective terms. We generate a UNL (Universal Networking Language) graph for the input *text*. Rules are applied on the graph to extract relevant terms. The sentiment expressed in these terms is used to figure out the overall sentiment of the *text*. Results on binary sentiment classification have shown promising results.

5.1 Introduction

Many works in sentiment analysis try to make use of shallow processing techniques. The common thing in all these works is that they merely try to identify sentiment-bearing expressions as shown by [Ruppenhofer and Rehbein, 2012]. No effort has been made to identify which expression actually contributes to the overall sentiment of the text. In [Mukherjee and Bhattacharyya, 2012] these expressions are given weight-age according to their position w.r.t. the discourse elements in the *text*. But it still takes into account each expression.

Semantic analysis is essential to understand the exact meaning conveyed in the *text*. Some words tend to mislead the meaning of a given piece of *text* as shown in the previous example. WSD (Word Sense Disambiguation) is a technique which can been used to get the right sense of the word. [Balamurali, Joshi, and Bhattacharyya, 2011] have made use of WordNet synsets for a

supervised sentiment classification task. [Martın-Wanton, Balahur-Dobrescu, Montoyo-Guijarro, and Pons-Porrata, 2010] and [Rentoumi, Giannakopoulos, Karkaletsis, and Vouros, 2009] have also shown a performance improvement by using WSD as compared to word-based features for a supervised sentiment classification task. In [Saif, He, and Alani, 2012], semantic concepts have been used as additional features in addition to word-based features to show a performance improvement. Syntagmatic or structural properties of text are used in many NLP applications like machine translation, speech recognition, named entity recognition, etc. A clustering based approach which makes use of syntactic features of text has been shown to improve performance in [AR, Bhattacharyya, and Haffari, 2009]. Another approach can be found in [Mukherjee and Bhattacharyya, 2012] which makes use of lightweight discourse for sentiment analysis. In general, approaches using semantic analysis are expensive than syntax-based approaches due to the shallow processing involved in the latter. As pointed out earlier, all these works incorporate all the sentiment-bearing expressions to evaluate the overall sentiment of the *text*. The fact that not all expressions contribute to the overall sentiment is completely ignored due to this. Our approach tries to resolve this issue. To do this, we create a UNL graph for each piece of *text* and include only the relevant expressions to predict the sentiment. Relevant expressions are those which satisfy the rules/conditions. After getting these expressions, we use a simple dictionary lookup along with attributes of words in a UNL graph to calculate the sentiment.

In the next section, we will go through some of the related work in this direction.

5.2 Related Work

There has been a lot of work on using semantics in sentiment analysis. [Saif, He, and Alani, 2012] have made use of semantic concepts as additional features in a word-based supervised sentiment classifier. Each entity is treated as a semantic concept e.g. *iPhone, Apple, Microsoft, MacBook, iPad, etc.*. Using these concepts as features, they try to measure their correlation with positive and negative sentiments. In [Verma and Bhattacharyya, 2009], effort has been made to construct document feature vectors that are sentiment-sensitive and use world knowledge. This has been achieved by incorporating sentiment-bearing words as features in document vectors. The use of WordNet synsets is found in [Balamurali, Joshi, and Bhattacharyya, 2011], [Rentoumi, Giannakopoulos, Karkaletsis, and Vouros, 2009] and [Martın-Wanton, Balahur-Dobrescu, Montoyo-Guijarro, and Pons-Porrata, 2010]. The one thing common with these approaches is that they make use of shallow semantics. An argument has been made in [Choi and Cardie, 2008] for determining the polarity of a sentiment-bearing expression that words or constituents within the expression can interact with each other to yield a particular overall polarity. Structural inference motivated by compositional semantics has been used in this work. This work shows use of deep semantic information for the task of sentiment classification. A novel use of

semantic frames is found in [Ruppenhofer and Rehbein, 2012]. As a step towards making use of deep semantics, they propose SentiFrameNet which is an extension to FrameNet. A semantic frame can be thought of as a conceptual structure describing an event, relation, or object and the participants in it. It has been shown that potential and relevant sentiment bearing expressions can be easily pulled out from the sentence using the SentiFrameNet. All these works try to bridge the gap between rule-based and machine-learning based approaches but except the work in [Ruppenhofer and Rehbein, 2012], all the other approaches consider all the sentiment-bearing expressions in the text.

5.3 Use of Deep Semantics

Before devising any solution to a problem, it is advisable to have a concise definition of the problem. Let us look at the formal definition of the sentiment analysis problem as given in [Liu, 2010]. Before we do that, let us consider the following review for a movie, *"1) I went to watch the new James Bond flick, Skyfall at IMAX which is the best theater in Mumbai with my brother a month ago. 2) I really liked the seating arrangement over there. 3) The screenplay was superb and kept me guessing till the end. 4) My brother doesnt like the hospitality in the theater even now. 5) The movie is really good and the best bond flick ever."* This is a snippet of the review for a movie named Skyfall . There are many entities and opinions expressed in it. 1) is an objective statement. 2) is subjective but is intended for the theater and not the movie. 3) is a positive statement about the screenplay which is an important aspect of the movie. 4) is a subjective statement but is made by the authors brother and also it is about the hospitality in the theater and not the movie or any of its aspects. 5) reflects a positive view of the movie for the author. We can see from this example that not only the opinion but the opinion holder and the entity about which the opinion has been expressed are also very important for sentiment analysis. Also, as can be seen from 1), 4) and 5) there is also a notion of time associated with every sentiment expressed. Now, let us define the sentiment analysis problem formally as given in [Liu, 2010].

A direct opinion about the object is a quintuple $< o_j, f_{jk}, oo_{ijkl}, h_i, t_l >$*, where* o_j *is the the object,* f_{jk} *is the feature of the object* o_j*,* oo_{ijkl} *is the orientation or polarity of the opinion on feature* f_{jk} *of object* o_j*,* h_i *is the opinion holder and* t_i *is the time when the opinion is expressed by* h_i*.*

As can be seen from the formal definition of sentiment analysis and the motivating example, not all sentiment-bearing expressions contribute to the overall sentiment of the *text*. To solve this problem, we can make use of semantic roles in the *text*. Semantic role is the underlying relationship that the underlying participant has with the main verb. To identify the semantic roles, we make use of UNL in our approach.

5.3.1 UNL (Universal Networking Language)

UNL is declarative formal language specifically designed to represent semantic data extracted from natural language texts. In UNL, the information is represented by a semantic network, also called UNL graph. UNL graph is made up of three discrete semantic entities, Universal Words, Universal Relations, and Universal Attributes. Universal Words are nodes in the semantic network, Universal Relations are arcs linking UWs, and Universal attributes are properties of UWs. To understand UNL better, let us consider an example. UNL graph for *"I like that bad boy"* is as shown in the figure.

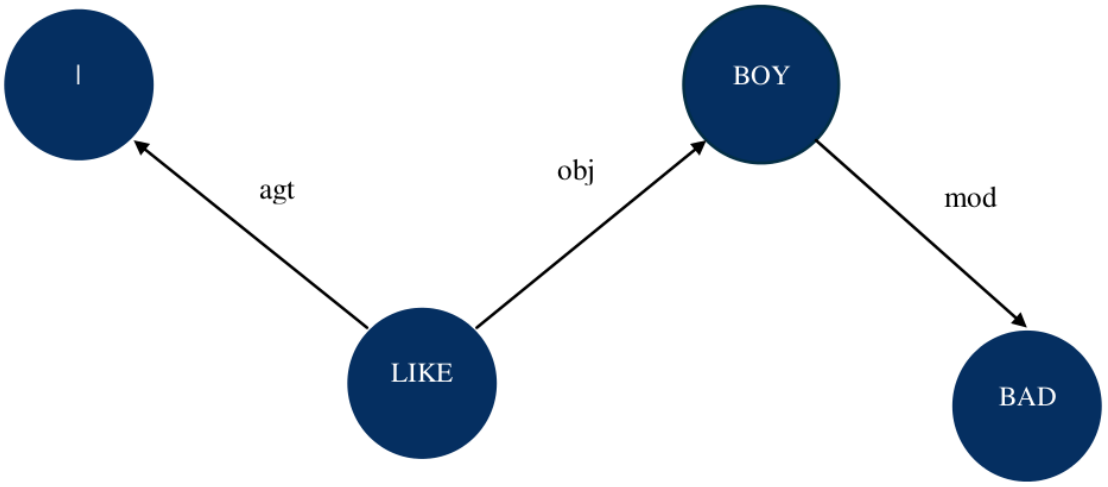


Figure 5.1 UNL Example

Here, *"I"*, *"like"*, *"bad"*, and *"boy"* are the UWs. *"agt"* (agent), *"obj"* (patient), and *"mod"* (modifier) are the Universal Relations. Universal attributes are the properties associated with UWs which will be explained as and when necessary with the rules of our algorithm.

UNL relations

Syntax of a UNL relation is as shown below,

$$< rel >:< scope >< source >;< target >$$

Where, $< rel >$ is the name of the relation, $< scope >$ is the scope of the relation, $< source >$ is the UW that assigns the relation, and $< target >$ is the UW that receives the relation

We have considered the following Universal relations in our approach,

1. *agt* relation : *agt* stands for agent. An agent is a participant in action that provokes a change of state or location. The *agt* relation for the sentence *"John killed Mary"* is *agt(killed , John)*. This means that the action of *killing* was performed by *John*.

2. *obj* relation : *obj* stands for patient. A patient is a participant in action that undergoes a change of state or location. The *obj* relation for the sentence *"John killed Mary"* is *obj(killed , Mary)*. This means that the patient/object of *killing* is *Mary*.

3. *aoj* relation : *aoj* stands for object of an attribute. In the sentence *"John is happy"*, the *aoj* relation is *aoj(happy , John)*.

4. *mod* relation : *mod* stands for modifier of an object. In the sentence *"a beautiful book"*, the *mod* relation is *mod(book , beautiful)*.

5. *man* relation : *man* relation stands for manner. It is used to indicate how the action, experience or process of an event is carried out. In the sentence *"The scenery is beautifully shot"*, the *man* relation is *man(beautifully , shot)*.

6. *and* relation : *and* relation is used to state a conjunction between two entities. In the sentence *"Happy and cheerful"*, the *and* relation is *and(Happy,cheerful)*.

5.3.2 Architecture

As shown in the *UNL* example, the modifier *"bad"* is associated with the object of the main verb. It shouldn't affect the sentiment of the main agent. Therefore, we can ignore the modifier relation of the main object in such cases. After doing that, the sentiment of this sentence can be inferred to be positive. The approach followed in the project is to first generate a UNL graph for the given input sentence. Then a set of rules is applied and used to infer the sentiment of the sentence. The process is shown in Figure 6.2. The UNL generator shown in the Figure 6.2 has been developed at CFILT.[1] Before, the given piece of text is passed on to the UNL generator, it goes through a number of pre-processing stages. Removal of redundant punctuations, special characters, emoticons, etc. are part of this process. This is extremely important because the UNL generator is not able to handle special characters at the moment. We can see that, the

[1] http://www.cfilt.iitb.ac.in/

performance of the overall system is limited by this. A more robust version of the UNL generator will certainly allow the system to infer the sentiment more accurately.

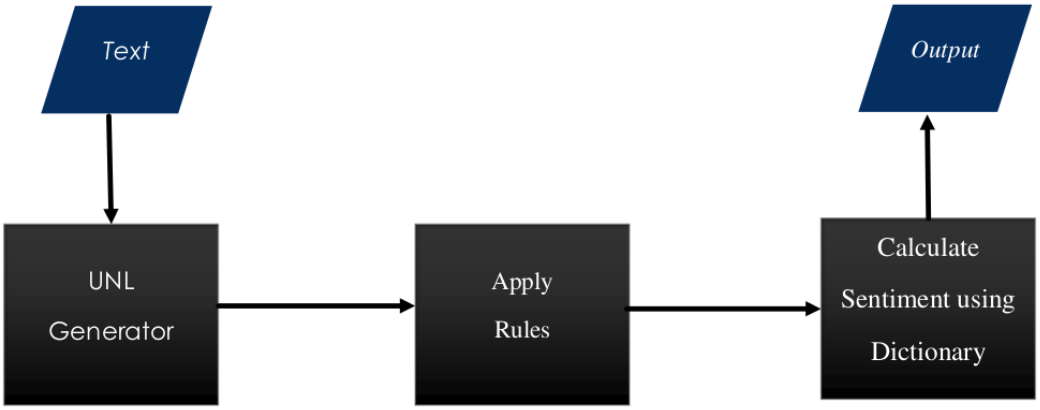


Figure 5.2 Architecture

5.3.3 Rules

There is a separate rule for each relation. For each UW (Universal word) considered, if it has a *@not* attribute then its polarity is reversed. Rules used by the system are as follows,

1. If a given UW is source of the *agt* relation, then its polarity is added to the overall polarity of the text. e.g., *"I like her"*. Here, the agt relation will be *agt (like , I)*. The polarity of like being positive, the overall polarity of the text is positive. e.g, *"I don't like her"*. Here the agt relation will be *agt (like@not , I)*. The polarity of like is positive but it has an attribute *@not* so its polarity is negative. The overall polarity of the text is negative in this case.

2. If a given UW is source or target of the *obj* relation and has the attribute *@entry* then its polarity is added to the overall polarity of the text. This rule merely takes into account the main verb of the sentence into account, and the it's is polarity considered. e.g., *"I like her"*, here the obj relation will be *obj (like@entry , her)*. The polarity of like being positive, the overall polarity of the text is positive

3. If a given UW is the source of the *aoj* relation and has the attribute *@entry* then its polarity is added to the overall polarity of the text. e.g., *"Going downtown tonight it will be amazing on the waterfront with the snow"*. Here, the *aoj* relation is *aoj (amazing@entry ,*

it). amazing has a positive polarity and therefore overall polarity is positive in this case.

4. If a given UW is the target of the *mod* relation and the source UW has the attribute *@entry* or has the attribute *@indef* then polarity of the target UW is added to the overall polarity of the text. e.g., *"I like that bad boy"*. Here, the aoj relation is *mod (boy , bad). bad* has a negative polarity but the source UW, boy does not have an *@entry* attribute. So, in this case negative polarity of bad is not considered as should be the case. e.g., *"She has a gorgeous face"*. Here, the mod relation is *mod (face@indef , gorgeous). gorgeous* has a positive polarity and face has an attribute *@indef*. So, polarity of gorgeous should be considered.

5. If a given UW is the target of the *man* relation and the source UW has the attribute *@entry* then polarity of the target UW is added to the overall polarity of the text. Or if the target UW has the attribute *@entry* then also we can consider polarity of the target UW. e.g., *"He always runs fast"*. Here, the *aoj* relation is *mod (run@entry , fast). fast* has a positive polarity and the source UW, run has the *@entry* attribute. So, in this case positive polarity of fast is added to the overall polarity of the sentence. Polarities of both the source and target UW of the *and* relation are considered.

6. In *"Happy and Cheerful"*, the *and* relation is *and(Happy, Cheerful). Happy* and *Cheerful*, both have a positive polarity, which gives this sentence an overall positive polarity.

The polarity value of each individual word is looked up in a dictionary of positive of negative words used is [Liu, 2010] After all the rules are applied, summation of all the calculated polarity values is done. If this sum is greater than 0 then it is considered as positive, and negative otherwise. This system is negative biased due to the fact that people often tend to express negative sentiment indirectly or by comparison with something good.

In the next section, we will explain the experimental setup, results obtained and subsequent discussion.

5.4 Experiments

Analysis was performed for monolingual binary sentiment classification task. The language used in this case was *English*. The comparison was done between 5 systems viz. System using words as features, WordNet sense based system as given in [Balamurali, Joshi, and Bhattacharyya, 2011], Clusters based system as described in [AR, Bhattacharyya, and Haffari, 2009], Discourse

rules based system as given in [Mukherjee and Bhattacharyya, 2012], and the UNL rule based system.

5.4.1 Datasets

Two polarity datasets were used to perform the experiments.

1. EN-TD: English Tourism corpus as used in [Ye, Zhang, and Law, 2009]. It consists of 594 positive and 593 negative reviews.
2. EN-PD: English Product (music albums) review corpus [Blitzer, Dredze, and Pereira, 2007]. It consists of 702 positive and 702 negative reviews.

For the WordNet sense, and Clusters based systems, a manually sense tagged version of the (EN-PD) has been used. Also, a automatically sense tagged version of (EN-TD) was used on these systems. The tagging in the later case was using an automated WSD engine, trained on a tourism domain [Balamurali, Khapra, and Bhattacharyya, 2013]. The results reported for supervised systems are based on 10-fold cross validation.

5.4.2 Results

System	**EN-TD**	**EN-PD**
Bag of Words	85.53	73.24
Synset-based	88.47	71.58
Cluster-based	**95.20**	79.36
Discourse-based	71.52	64.81
UNL rule-based	86.08	**79.55**

Table 5.1 Classification accuracy (in %) for monolingual binary sentiment classification

The results for monolingual binary sentiment classification task are shown in Table 5.1. The results reported are the best results obtained in case of supervised systems. The cluster based system performs the best in both cases. The UNL rule-based system performs better only than the bag of words and discourse rule based system. For EN-PD (music album reviews) dataset, the UNL based system outperforms every other system . These results are very promising for a rule-based system. The difference between accuracy for positive and negative reviews for the rule-based systems viz. Discourse rules based and UNL rules based is shown in Table 5.2. It can be seen that the Discourse rules based system performs slightly better than the UNL based

system for positive reviews. On the other hand, the UNL rules based system outperforms it in case of negative reviews by a huge margin.

	EN-TD		**EN-PD**	
System	**Pos**	**Neg**	**Pos**	**Neg**
Discourse rules	94.94	48.06	**92.73**	36.89
UNL rules	**95.72**	**76.44**	90.75	**68.35**

Table 5.2 Classification accuracy (in %) for positive and negative reviews

5.4.3 Discussion

The UNL generator used in this case is the bottleneck in terms of performance. Also, it makes use of the standard NLP tools viz. parsing, co-reference resolution, etc. to assign the proper semantic roles in the given *text*. It is well known fact that these techniques work properly only on structured data. The language used in the reviews present in both the datasets is unstructured in considerable number of cases. The UNL generator is still in its infancy and cannot handle *text* involving special characters. Due to these reasons, a proper UNL graph is not generated in some cases. Also, it is not able to generator proper UNL graphs for even well structured sentences like *"It is not very good"*. In this case, the UW, *good* should have an attribute *@not* which implies that it is used in a negative sense. On the contrary, the UNL generator does not assign any such attribute to it, leading to incorrect classification. As a result of all these things, the classification accuracy is low.

Negative reviews have certain characteristics which make them difficult to classify. Some of them are listed below.

1. **Relative opinions containing positive words**
   (a) Sentences like *"It is good but could be better"* are very common in negative reviews.
   (b) A user in a review about a certain travel destination says, *"Came here just two days ago. Lucky to have got our flight back home"*
2. **Some negative reviews are just to raise caution**
   A negative review about *Las Vegas* contained the sentence, *"Do not go alone to Las Vegas if you are a female"*
3. **Mixed opinions**
   Some reviews contain mixed opinions. For example, *"Beautiful Architecture. Expensive food"*.

4. **A sense of time**

   We stated the importance of time in sentiment analysis in Section 5.3. A perfect example which shows the importance of time is, *"A place I used to love. Not anymore"*. One more example is *"The best city in the world, 'Once upon a time'."*.

5. **Lack of domain knowledge**

   Some reviews require domain knowledge to understand their meaning. Consider the sentence *"They gave us a Queen-size bed"*. Here, the the phrase *"Queen-size bed"* refers to a small bed. People usually refer to beds having sufficient space as *"King-size bed"*.

6. **Domain dependence of sentiment**

   Sentiment is domain dependent as given in [Liu, 2010]. The adjective *cheesy* might be positive for a food item but is definitely negative in cases like *"The place is full of cheesy shows"*.

7. **Comparison with other entities**

   Reviewers often criticize a place/thing by praising other places/things. In the EN-TD dataset, negative reviews were full of sentences like *"If you want dirt, go to L.A. . If you want peace, go to Switzerland."*.

8. **Sarcasm**

   Sarcasm is the most notorious problems in *SA*. *"I love New York especially the 'Lovely brown fog"'* and *"I adore the 'phony fantasy-land' that Vegas is"* are examples of sarcasm. Sarcasm is a very difficult problem to tackle. Some related works can be found in [Carvalho, Sarmento, Silva, and de Oliveira, 2009] and [González-Ibánez, Muresan, and Wacholder, 2011].

9. **New notations**

   During our error analysis, we observed sentences like, *"I love it ?!?!"*, *"Too $$$$"*, *"No value for $$$$"*. In these examples the negative sentiment is clear but difficult to detect.

In some cases, the reviewers make use of their native language and expressions. This is a big problem for the task of monolingual sentiment classification. From this discussion, it is clear that two major points of concern are unstructured language and hidden sentiment.

SUMMARY

In the beginning, we expressed the motivation behind this approach. After that, related work was discussed. Then we explained the approach to use deep semantics in detail. The system using deep semantics was compared with some state of the art systems for the sentiment classification task with two datasets. This was followed up by some interesting observations and error analysis.

In the next chapter, we conclude and comment on some future work.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

Topical n-gram model can be used for the binary sentiment classification problem. The motivation behind this approach was to use not only words but also phrases to classify documents. The model was trained with documents containing only subjective and negation words and bigrams formed by the combination of these. It also made use of rules to assign topics to words and bigrams in the intialization stage of estimation. Results using a prior show statistically significant improvement over the JST model. The paper also shows the use of LDA for resource generation, prompted by observations made during error analysis. The system shows marginal improvement in accuracy after using the resources generated by this technique.

Using deep semantics for sentiment analysis of structured data increases classification accuracy sentiment analysis. A semantic role labeling method through generation of a UNL graph was used to do this. The main motivation behind this research was the fact that not all sentiment bearing expressions contribute to the overall sentiment of the *text*. The approach was evaluated on two datasets and compared with successful previous approaches which don't make use of deep semantics. The system underperformed all the supervised systems but showed promise by yielding better results than the other rule-based approach. Also, in some cases the performance was very close to the other supervised systems. The system works well on sentences where are inherently complex and difficult for sentiment analysis as it makes use of semantic role labeling. Any rule based system can never be exhaustive in terms of rules. We always need to add new rules to improve on it. In some case, adding new rules might cause side-effects. In this case, as the rules are intuitive, adding of new rules will be easy. Also, analysis of the results hints at some ways to tackle specific problems effectively.

6.2 Future work

6.2.1 Short term

The system using topical n-grams model is focused on bigrams at the moment. It can be extended to handle n-grams by adding rules to detect and assign topics. Instead of adding rules, we can make use of machine learning techniques to detect the subjective nature of a phrase. For the *UNL* system, Adding more rules to the system will help to improve the system. Language gets updated almost daily, we plan to update our dictionary with these new words and expressions to increase the accuracy. Also, we plan to replace the UNL system with a dependency parsing system and apply rules similar to the ones described in this work.

6.2.2 Medium term

Use of topic models and semantic information, to perform the aspect based sentiment analysis. We can identify the various terms using deep semantics and their associated sentiment can be found out using rules. The sentiment for each term can be used to find out the sentiment of each aspect by the use of topic models.

6.2.3 Long term

A combined model of topic models and semantics for binary sentiment classification.

References

Klaus R Scherer, Tanja Bänziger, and Etienne B Roesch. *Blueprint for Affective Computing: A Sourcebook and Manual*. Oxford University Press, 2010.

Klaus R Scherer. What are emotions? and how can they be measured? *Social science information*, 44(4):695–729, 2005.

Bing Liu. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2:568, 2010.

Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67, 1999.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12, 2009.

Subhabrata Mukherjee, Pushpak Bhattacharyya, and AR Balamurali. Sentiment analysis in twitter with lightweight discourse analysis. 2010.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

Andrea Esuli and Fabrizio Sebastiani. Determining term subjectivity and term orientation for opinion mining. In *Proceedings of EACL*, volume 6, pages 193–200, 2006.

Apache Lucene, 2013. URL http://lucene.apache.org/.

SentiWordNet, 2013. URL http://sentiwordnet.isti.cnr.it/.

Jay M Ponte and W Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM, 1998.

Rasmus E Madsen, David Kauchak, and Charles Elkan. Modeling word burstiness using the dirichlet distribution. In *Proceedings of the 22nd international conference on Machine learning*, pages 545–552. ACM, 2005.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Gregor Heinrich. Parameter estimation for text analysis. *Web: http://www.arbylon.net/publications/text-est.pdf*, 2005.

Brian Walsh. Markov chain monte carlo and gibbs sampling. 2004.

Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM, 2009.

dmoz open directory project, 2013. URL http://www.dmoz.org/.

Dmitriy Bespalov, Bing Bai, Yanjun Qi, and Ali Shokoufandeh. Sentiment classification based on supervised latent n-gram analysis. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 375–382. ACM, 2011.

Peter Turney and Michael L Littman. Unsupervised learning of semantic orientation from a hundred-billion-word corpus. 2002.

Koji Eguchi and Victor Lavrenko. Sentiment retrieval using generative models. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 345–354. Association for Computational Linguistics, 2006.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, pages 171–180. ACM, 2007.

Min Zhang and Xingyao Ye. A generation model to unify topic relevance and lexicon-based sentiment for opinion retrieval. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 411–418. ACM, 2008.

Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 375–384. ACM, 2009.

Yohan Jo and Alice H Oh. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM, 2011.

Ellen Riloff, Janyce Wiebe, and William Phillips. Exploiting subjectivity classification to improve information extraction. In *Proceedings of the National Conference On Artificial Intelligence*, volume 20, page 1106. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

Charlotta Engström. Topic dependence in sentiment classification. *Unpublished MPhil Dissertation. University of Cambridge*, 2004.

Xuerui Wang, Andrew McCallum, and Xing Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 697–702. IEEE, 2007.

Xuerui Wang and Andrew McCallum. A note on topical n-grams. 2005.

Josef Ruppenhofer and Ines Rehbein. Semantic frames as an anchor representation for sentiment analysis. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, pages 104–109. Association for Computational Linguistics, 2012.

Subhabrata Mukherjee and Pushpak Bhattacharyya. Sentiment analysis in twitter with lightweight discourse analysis. In *COLING*, pages 1847–1864, 2012.

AR Balamurali, Aditya Joshi, and Pushpak Bhattacharyya. Harnessing wordnet senses for supervised sentiment classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1081–1091. Association for Computational Linguistics, 2011.

Tamara Martın-Wanton, Alexandra Balahur-Dobrescu, Andres Montoyo-Guijarro, and Aurora Pons-Porrata. Word sense disambiguation in opinion mining: Pros and cons. *Special issue: Natural Language Processing and its Applications*, 119:358, 2010.

Vassiliki Rentoumi, George Giannakopoulos, Vangelis Karkaletsis, and George A Vouros. Sentiment analysis of figurative language using a word sense disambiguation approach. In *Proceedings of the International Conference RANLP*, pages 370–375, 2009.

Hassan Saif, Yulan He, and Harith Alani. Semantic sentiment analysis of twitter. In *The Semantic Web–ISWC 2012*, pages 508–524. Springer, 2012.

Kashyap Popat2 Balamurali AR, Pushpak Bhattacharyya, and Gholamreza Haffari. The haves and the have-nots: Leveraging unlabelled corpora for sentiment analysis. 2009.

Shitanshu Verma and Pushpak Bhattacharyya. Incorporating semantic knowledge for sentiment analysis. *Proceedings of ICON*, 2009.

Yejin Choi and Claire Cardie. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 793–801. Association for Computational Linguistics, 2008.

Qiang Ye, Ziqiong Zhang, and Rob Law. Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. *Expert Systems with Applications*, 36(3):6527–6535, 2009.

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447, 2007.

AR Balamurali, Mitesh M Khapra, and Pushpak Bhattacharyya. Lost in translation: viability of machine translation for cross language sentiment analysis. In *Computational Linguistics and Intelligent Text Processing*, pages 38–49. Springer, 2013.

Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio de Oliveira. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-). In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56. ACM, 2009.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics, 2011.