Indian Institute of Technology, Bombay

M.Tech Project Stage II

Different approaches to Sentiment Analysis: Topic modeling and Semantics

Author:
Nikhilkumar Jadhav

Supervisor:
Dr. Pushpak Bhattacharya

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

in the

Computer Science and Engineering Department

June 2014

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 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 role of sentiment analysis in information retrieval in particular. *Sentiment Analysis* has lot of applications as discussed. But, the assumption made before applying this technique in many cases is that subjective data is available. This assumption is unrealistic. Subjective text has to be retrieved from the web. But, most of the algorithms in *Information Retrieval* are designed to fetch information relevant to a specific topic or a topic set. If *Subjectivity Analysis* is combined with *IR* then we can fetch subjective *text*. Also, retrieval of *text* of a specific sentiment is required in many applications. Thus, combining *SA* with *IR* to serve the needs of many applications is the motivation behind this report.

1.2 Problem Statement

The aim of this project is to aid retrieval of subjective *text* of the desired sentiment by making use of *Sentiment Analysis* in *Information Retrieval*. To achieve this a joint model of topic and sentiment has to be designed, implemented and compared with other existing approaches. Also, the use of this model in sentiment classification of *text* has to be evaluated.

1.3 Contribution

Following contributions have been made till now:

1. Implemented two systems using [Lucene, 2013], [SentiWordNet, 2013], and [tagger, 2013] for sentiment aware information retrieval to get a gist of the problem.
2. Studied *LDA*, Gibbs Sampling, and Inferencing using *LDA*.
3. Evaluated *LDA* using implementation in [Mallet, 2002].
4. Studied several joint sentiment topic models.

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*. Chapter 4 discusses two models which combine sentiment and topics. The pros and cons of these models have been discussed. Chapter 5 describes two systems which make use of sentiment analysis in information retrieval. Also, an experiment to evaluate *LDA* is described. 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 has been 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

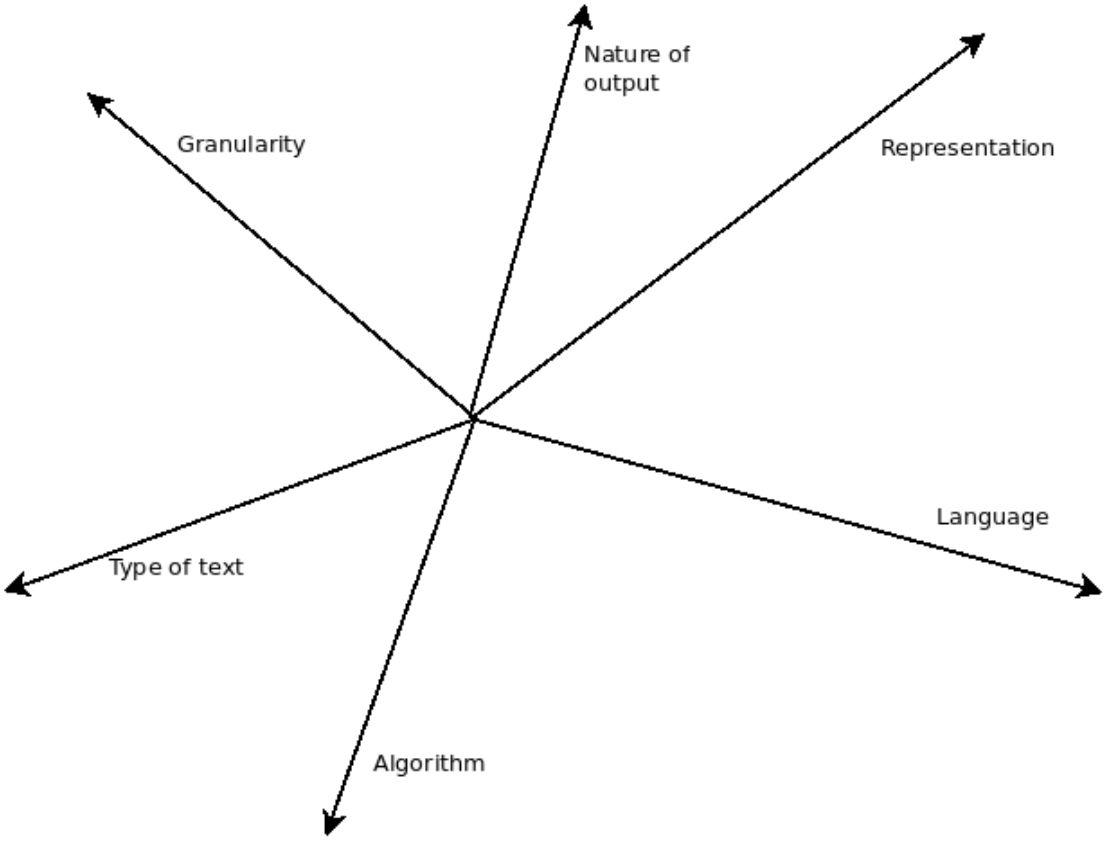
"A direct opinion (opinion about the object) is a quintuple $(o_j, f_{jk}, oo_{ijkl}, h_i, t_l)$*, where* o_j *is an object,* f_{jk} *is a feature of the object* o_j*,* oo_{ijkl} *is the orientation or polarity of the opinion on feature* f_{jk} *of object* o_j*,* h_i *is the opinion holder and* t_l *is 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

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,...,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 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,

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

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

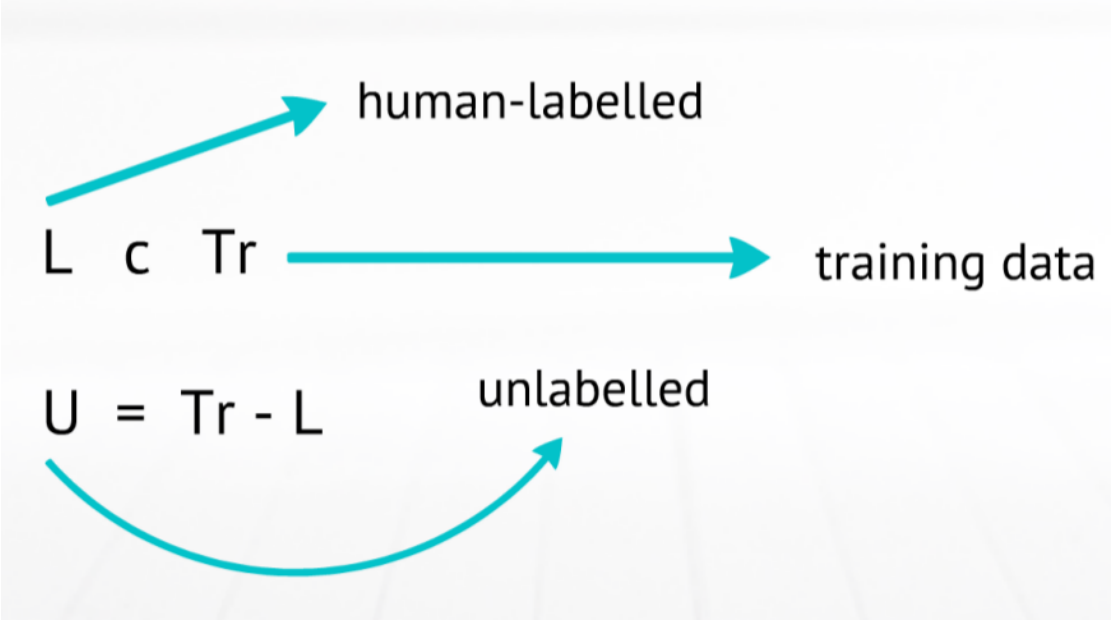


Figure 2.1 Training Data in semi-supervised learning

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.

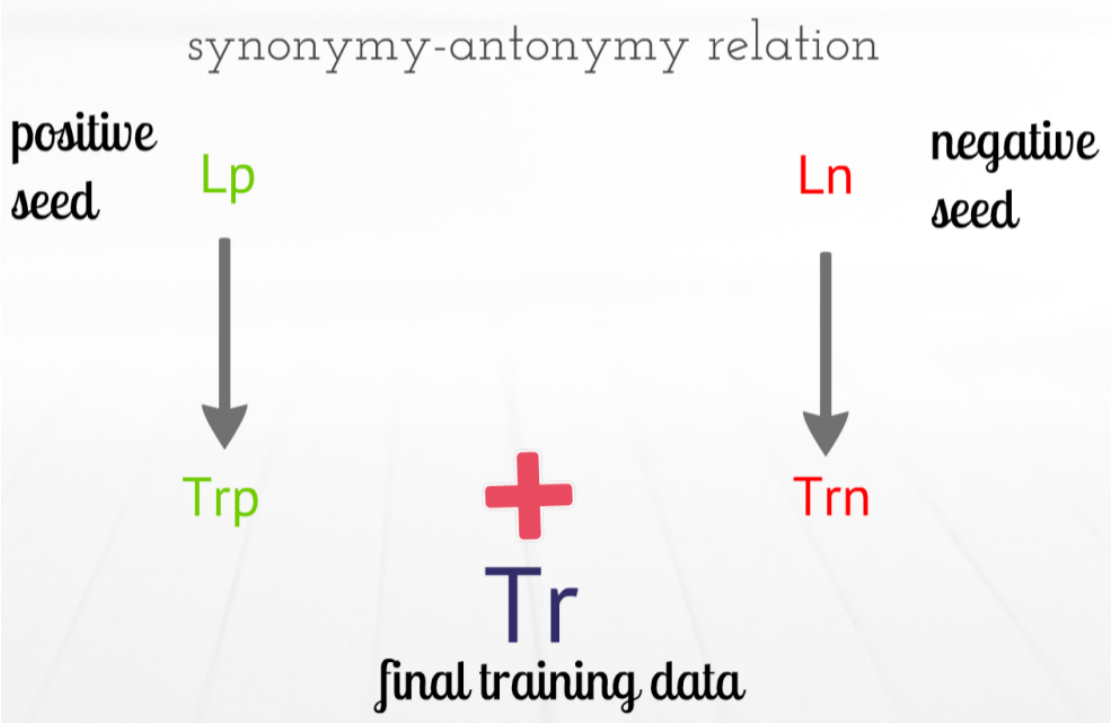


Figure 2.2 Using synonymy-antonymy relation for labeling unlabeled data

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.

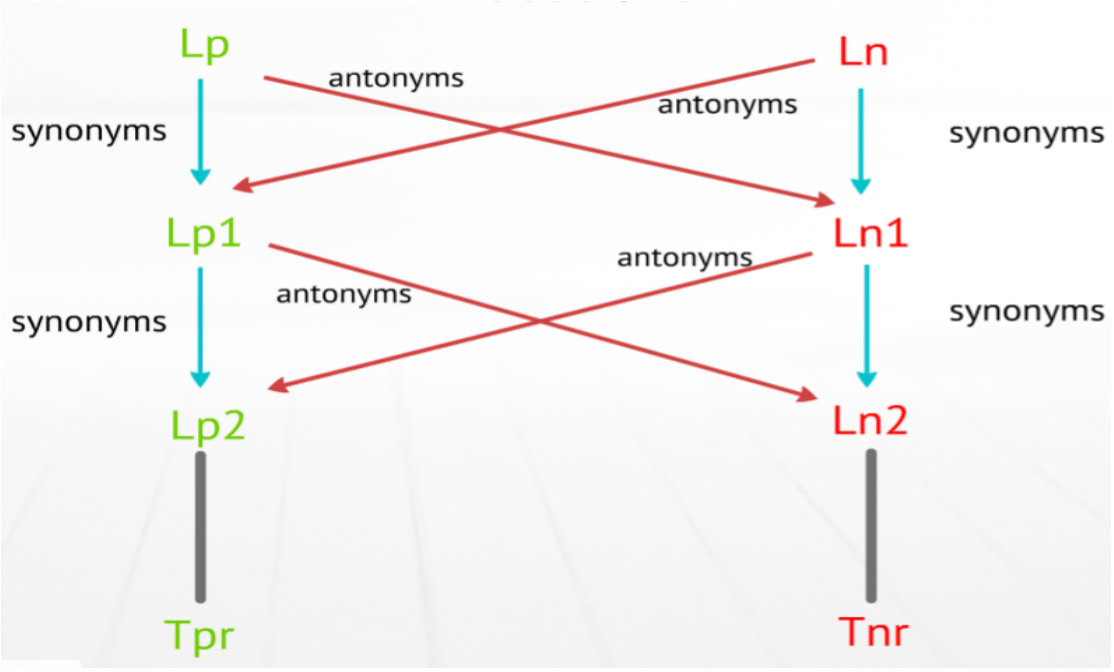


Figure 2.3 Procedure for labeling unlabeled data using synonymy-antonymy relation

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 and the focus on corpus models, mainly *LDA*.

Chapter 3

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 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]

3.2 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.

3.2.1 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 $Pr(x \mid \phi)$.

3.2.2 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:

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

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

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

3.2.3 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.

3.2.4 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.

3.2.5 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:

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

3.2.6 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.

3.3 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].

3.3.1 Bayesian Network for LDA

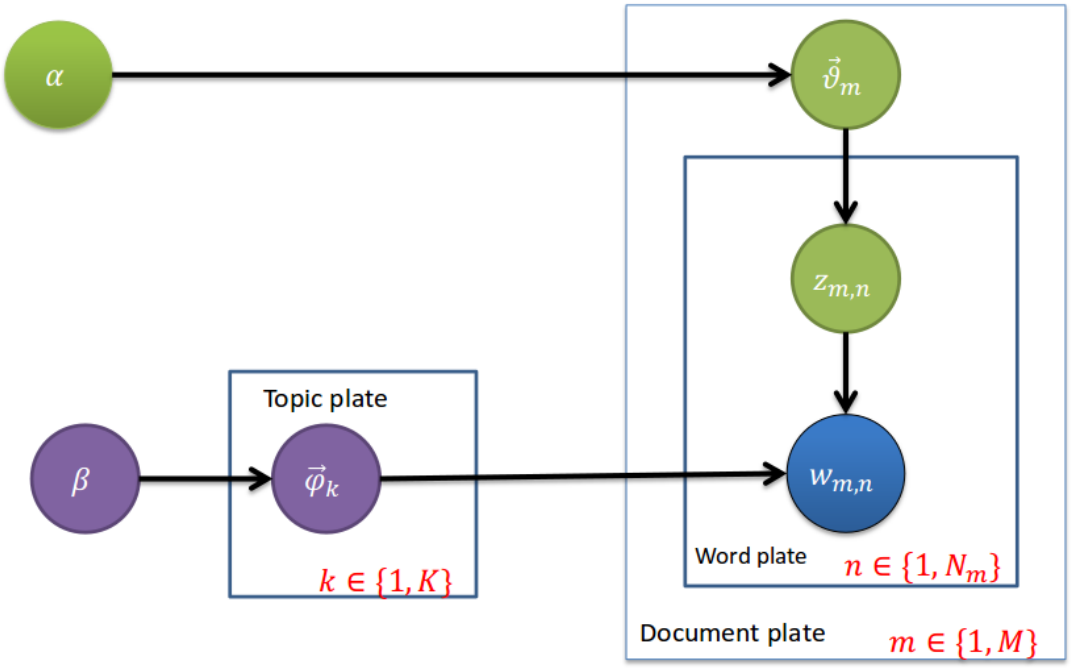


Figure 3.1 Bayesian Network for Latent Dirichlet Allocation

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.

3.3.2 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
```

3.3.3 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.`

3.3.4 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

3.3.5 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,

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

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 come up with a useful generative model which can model both topics and sentiments simultaneously. The unsupervised nature of this approach will make it very useful for many applications. In the next chapter we focus on some existing work in this direction.

Chapter 4

Joint Modeling of Sentiment and Topic

4.1 Modeling for Sentiment Aware Information Retrieval

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.

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.

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:

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

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.

4.2 Joint Sentiment-Topic modeling (JST)

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

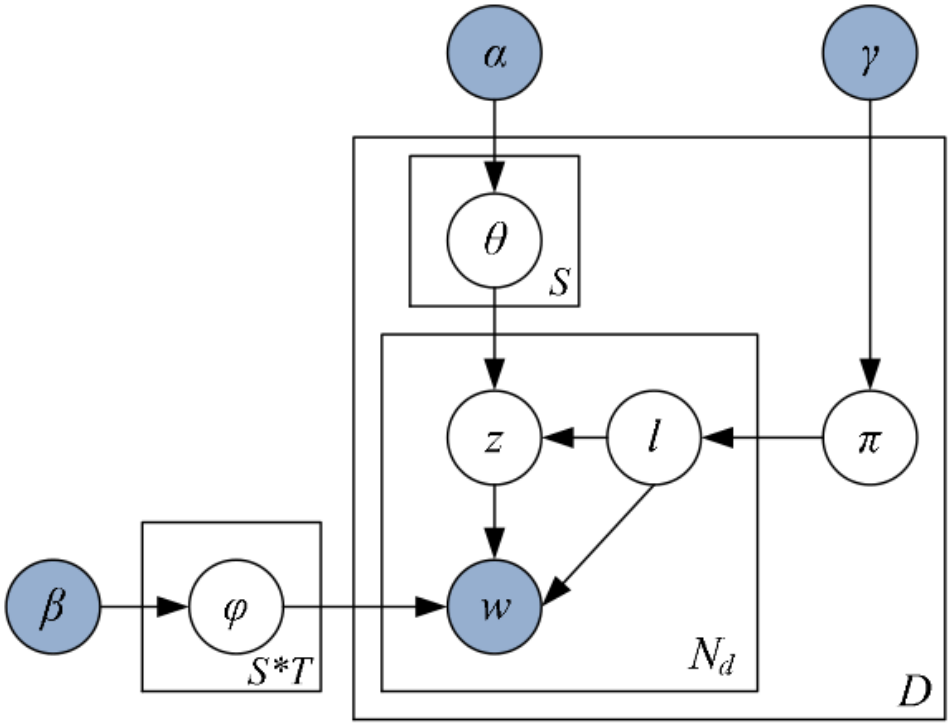


Figure 4.1 Joint Sentiment Topic Model

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.

Summary

In this chapter we discussed two models which combine sentiment and topics. Both the systems have shown high competence in classification tasks.

In the next chapter, we discuss the work done so far in this project. Two systems were developed to make use of *Sentiment analysis* in *information retrieval*. Also, the evaluation of *LDA* has been performed.

Chapter 5

Experiments

In this chapter we will be discussing about two systems implemented to use *sentiment analysis* in *information retrieval*. Also, *LDA* was evaluated to test the accuracy of unsupervised approaches in general.

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.

5.1 Indexing followed by Sentiment Analysis

5.1.1 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.

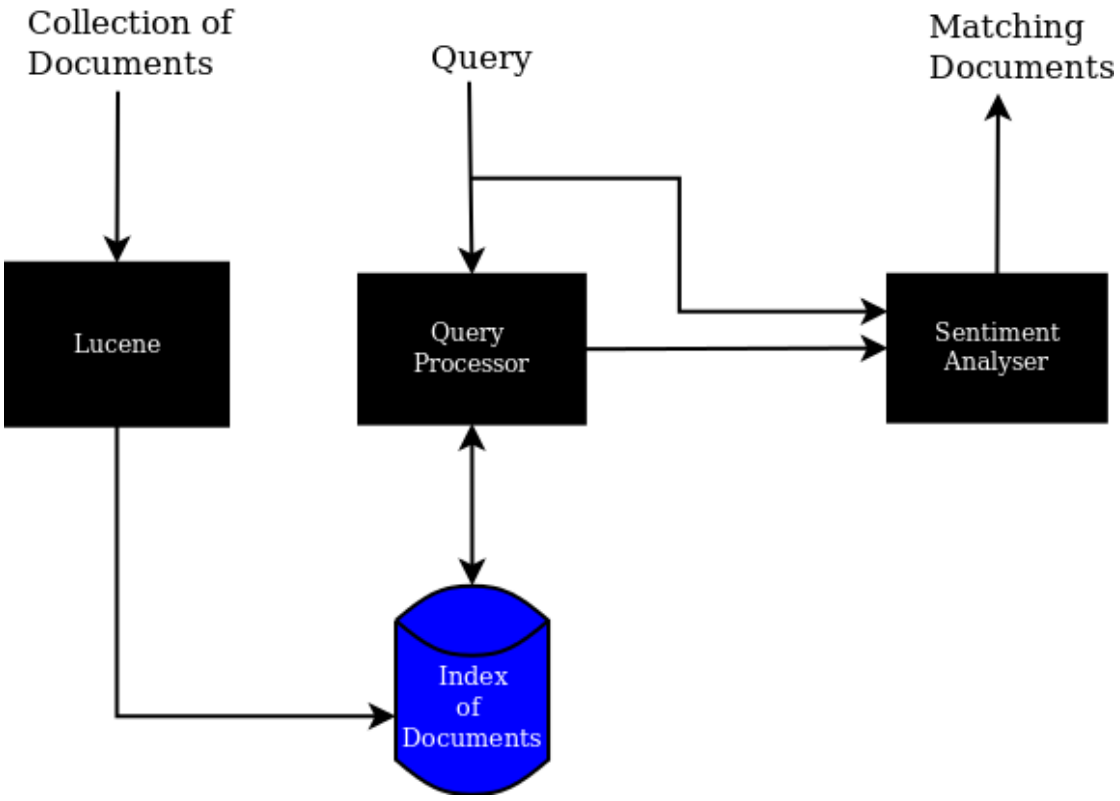


Figure 5.1 Indexing Followed by Sentiment Analysis

5.2 Encoding sentiment in the Index

5.2.1 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.

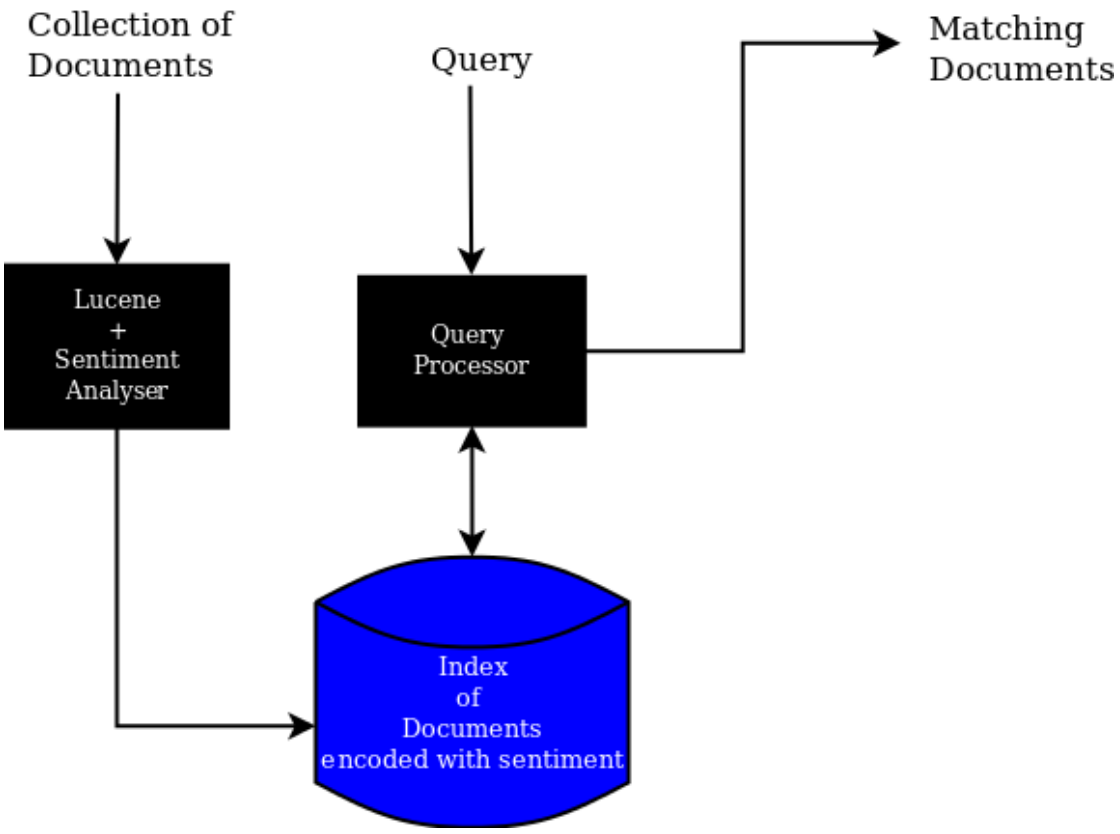


Figure 5.2 Encoding Sentiment in Index

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.

5.3 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 using 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 5.1 Number of files per topic

Average accuracy	20.867

Table 5.2 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 5.3 Accuracy for each testing fold

Average accuracy	62.6

Table 5.4 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 5.5 Accuracy for each testing fold (Till 3rd rank)

Average accuracy	51.7

Table 5.6 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 5.7 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

In this chapter, we discussed two systems that implement sentiment analysis in information retrieval using a procedural approach. Also, the pros and cons of these systems have been discussed. Also, the evaluation of *LDA* has been discussed.

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

Chapter 6

Conclusions and Future Work

6.1 Conclusion

In this report we have learned that a wide variety of approaches have been used for detecting sentiment in *text*. Supervised, Unsupervised and semi-supervised methods have been explored to solve this difficult problem. Due the unstructured nature of the *text* used for sentiment analysis, standard *NLP* tools cannot be used directly. This has lead many techniques to make use of bag-of-words like models which makes strong independence assumptions. The accuracy of bag-of-words model is less than desirable. So, this model is augmented with information about coherency, discourse, and pragmatics if it is present in the *text*. Graph based formulation has become really important to encode this information in the feature vector as it is very difficult to encode it in the standard *ML* techniques discussed. Domain dependence of sentiment also plays a very crucial role in sentiment analysis. A complete reversal of sentiment orientation is observed in some cases. Problems like sarcasm and thwarting still persist and no clear solution is present at the moment.

Information Retrieval is mostly concerned with factual data. Using subjectivity analysis, subjective *text* can be filtered out. This can remove *text* containing opinion of the author. But in some cases, retrieval of subjective text is very important. For example, you might want to fetch all reviews related to a product. This is called sentiment aware *IR*. In information retrieval, corpus modeling has been used extensively over the years to fetch documents related to a specific topics. The topics of interest are expressed by the words in the query. Topic modeling approach can be extended to model the sentiment of the *text*. When these two models are combined, not only does it help in fetching *text* pertaining to certain topics but also of the desired sentiment. Though these models make strong independence assumptions, their combination helps to model topic dependence of sentiment by encoding co-occurrence between sentiment and topic words.

Sentiment Analysis and *Information Retrieval* can be combined to achieve very good results in sentiment aware information retrieval.

Joint modeling of a sentiment and topics can also be used for sentiment classification. The sentiment which has highest proportion will be the sentiment of the *text*.

6.2 Future work

6.2.1 Short term

Designing a new generative model for sentiment and topics which is more intuitive that *JST*. *JST* doesn't exactly depict the process of document generation. The fact that the topic chosen depends on the sentiment is not intuitive. So, designing a new model which addresses this problem is necessary.

6.2.2 Medium term

Implementing the model and using it for sentiment classification. Comparing it with existing approaches for sentiment classification.

6.2.3 Long term

Unstructured nature of the *text* mostly involved in sentiment analysis is a very big obstacle. This makes it difficult to take into account discourse and pragmatics. There is shortage of work which consider these important elements of textual data. Graph based solutions to these problems have been discussed. Most of the future work in *SA* will focus on how to integrate this information in the feature vector so that it results in better classification.

Strong independence assumptions as we saw might mislead the classifier. Instead of using the same type of corpus modeling used in many *IR* applications, effort should be made to embed discourse and pragmatics information into the model. This will improve the accuracy of sentiment aware *IR* systems.

References

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

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

Stanford POS tagger, 2013. URL http://nlp.stanford.edu/software/tagger.shtml.

Mallet, 2002. URL http://mallet.cs.umass.edu/topics.php.

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.

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.

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.

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.

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

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.

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/.