# A Unified Probabilistic Model for Aspect-Level Sentiment Analysis

by

Daniel Stantic

A Thesis

Presented to

The University of Guelph

In partial fulfilment of requirements

for the degree of

Master of Science

in

Computer Science

Guelph, Ontario, Canada

ABSTRACT

# A Unified Probabilistic Model for Aspect-Level Sentiment Analysis

Daniel Stantic
University of Guelph, 2016

Advisor:
Dr. Fei Song

In this thesis, we develop a new probabilistic model for aspect-level sentiment analysis based on POSLDA, a topic classifier that incorporates syntax modelling for better performance. POSLDA separates semantic words from purely functional words and restricts its topic modelling on the semantic words. We take this a step further by modelling the probability of a semantic word expressing sentiment based on its part-of-speech class and then modelling its sentiment if it is a sentiment word. We restructure the popular approach of topic-sentiment distributions within documents and add a few novel heuristic improvements. Our experiments demonstrate that our model produces results competitive to the state of the art systems. In addition to the model, we develop a multi-threaded version of the popular Gibbs sampling algorithm that can perform inference over 1000 times faster than the traditional implementation while preserving the quality of the results.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

With the explosive popularity of the World Wide Web the available platforms for expressing opinions have proliferated exponentially. For any given product, political topic, celebrity or anything that can be evaluated subjectively, an Internet user is likely to find opinions on that topic. The opinions are often detailed; expressed not just on the overall topic but on specific aspects of that topic. For example, a user writing a review of a camera can talk about the picture quality, battery life, ease of use, etc. Reviews expressed in such detail give consumers valuable input when they are shopping for a specific type of product. They also give valuable feedback to the companies that produce and/or sell products being discussed and to public figures seeking to gauge public opinion. The value and power of the content generated on these platforms is indisputable.

Being able to extract useful information from all that raw data, however, is not always a trivial task. A popular product can have hundreds of reviews and if we look at all reviews for a certain type of product we could be facing thousands. Even though most review sites present an overall rating it may not be enough for a user who cares greatly about specific aspects of the product and not so much about others. A user who is looking to go on a cheap road-trip for example might care more about the price and location of a hotel whereas a business traveller will most likely care more about the business services. In order to extract that information users still have to sift through all the reviews manually.

Similarly, manufacturers of the products that are subjects of these reviews have to invest a lot of time to make sense of the data. Worst yet, for a public figure to read through thousands or even tens of thousands of opinions generated on social media sites and other forums is nearly impossible.

The problem of extracting topic aspects, their relative importance (the weights) and the sentiments expressed on those aspects is commonly called Aspect-Level Sentiment Analysis and has received a great amount of attention over the recent years. At their core these are all text classification tasks although the methods used in each task are quite different. Recent research has shown that one of the ways to improve the accuracy and precision of these tasks is to take advantage of any part-of-speech (POS) information available for the given text. In this thesis we present a novel probabilistic model for sentiment analysis while modelling syntax in unstructured text.

## 1.1 Applications

Public and consumer opinion has been the driving force for many of the decisions we make both as consumers and providers of goods and services. Many of the popular shopping sites such as Amazon, Expedia, TigerDirect and others provide the opportunity for their customers to rate the products they purchased and write free-text reviews. The information in these reviews is very useful to other shoppers as well as the retailers and manufacturers of those products. However, much more can be done with it with the right tools.

1. Aspect Sentiment Detection and Summarization

   As we mentioned before, most ratings given on shopping web sites are on the overall satisfaction or, at best, on broad categories applied to all products such as value, quality, features, etc. and not on aspects specific to that product. A shopper that cares about some specific features of a product still has to read through all the reviews to get a sense of what people think about those features. It is a tedious and time consuming process. Moreover, the shopper might only read the first several reviews thinking that they are representative of the rest of them when in fact the rest might contain different information. An automated way of parsing the review data and extracting this information would be much more efficient and valuable. The shopper can then be presented with exactly the information he needs at a glance.

2. Aspect Ranking Search

   If someone wants to find a product that is the best in the market when it comes to a specific feature he has to either go through the same exercise we mentioned in the previous point or, if he is lucky, he might find a web page where someone has done all the work and provided a ranking. Even though today's search engines are getting quite sophisticated there is still no easy way to retrieve this type of information. However, if a search engine had capabilities to do aspect-level sentiment analysis, a user would be able to enter search queries such as “camera with best picture quality and lowest price” to get the desired information.

3. Marketing

   The information contained in user reviews is valuable not just to consumers but to manufacturers and retailers as well. Traditionally, manufacturers have been resorting to marketing surveys and focus groups to get a sense of what their consumers like or dislike about their products. Such endeavours are very costly and usually capture only a small fraction of the consumer opinion. The online user reviews on the other hand do not require any action from the retailer to collect once the infrastructure is in place. The reviews are also open to all of their consumers as opposed to a chosen few as is the case in focus groups. An automated way of summarizing the data in these reviews would give the manufacturers a fast, easy and low-cost way of gathering valuable consumer feedback. That information can also be used by the retailers as their sales staff is often asked to make recommendations based on specific user needs.

## 1.2 General Challenges

Sentiment analysis in general poses several challenges.

1. High Vocabulary Variability

   When performing topic classification we can safely assume that certain words occur more frequently for a given topic. For example, “player”, ”ball” and “score” are likely to occur often in sports articles but not so often in articles covering other topics. Our topic classification approach can then take advantage of that fact. However, when people are expressing sentiment they tend to vary their vocabulary. It is very rare to see a review that reads something like “The movie was good. The actors were good. The plot was good.”. The reviewers tend to pick more varied words to express their sentiments so we cannot rely on the frequencies of the words “good” or “bad” for example. It might be tempting to say that we could rely on the frequencies of all words that are equivalent to “good” and all words that are equivalent to “bad” but there is a problem with that approach which brings us to our next point.

2. Context Dependence

   Evaluative language can be highly context-dependent. Consider the following two evaluative statements:

> “The movie was unpredictable”
>
> and
>
> “The car steering is unpredictable”

The first one uses the word “unpredictable” to describe a movie. Generally, this is a good thing; unpredictable movies keep us interested in watching more. The second statement uses the same word to describe the steering mechanism of a car. An unpredictable steering mechanism would, of course, be a horrible thing as the driver would have no control over the car and would end up in frequent accidents. Even words that one would think would always mean good or always mean bad can sometimes end up meaning the opposite. Consider the following excerpt from a movie review [41]:

> “The slow, methodical way he spoke. I loved it! It made him seem more arrogant and even more evil.”

Most people would agree that arrogant and evil are bad personal characteristics. However, when we watch a movie we want the villain to be arrogant and evil and we want it to really stand out in the actor's performance. Context dependence is one of the biggest challenges in sentiment analysis.

3. Noise

While professionally written and published articles tend to stay on topic, free-form reviews by the general public contain considerably more noise. Consider the following movie review excerpt:

"My wonderful boyfriend took me to see this movie for our anniversary. It was terrible."

The review is about a movie but the reviewer has decided to provide additional context in which she describes the qualities of her boyfriend – something completely unrelated to the quality of the movie. Human beings have no problem distinguishing between movies and boyfriends but a computer algorithm can be thrown off by the extraneous text.

4. Subtlety

Sentiment is not always expressed through a sentence that clearly declares something as having a certain property. It is at times more subtle than that. For example, the following sentence:

"How can anyone sit through this movie?"

clearly (to humans) expresses a disappointment in the movie. However, the sentiment is expressed in the form of a question and uses no adjectives, adverbs, similes or metaphors. Algorithms relying on declarative statements have difficulty catching such subtle clues.

## 1.3 Proposed Solution

Our solution builds on the findings of other researchers and adds several novel changes. The parametric POSLDA model developed by Darling [9] can be seen as the starting point of our solution. Darling combined the LDA [7] topic model with a Bayesian HMM syntax model in such a way that words in a semantic POS class carry topic information while purely functional words do not. We then add a component that determines whether or not a word expresses a sentiment based on its POS class in the same way that Li at al. did for STDP [24]. Unlike their approach, our model does not require a POS tagger since it contains a syntax modelling component. We then add a sentiment analysis component similar to the one developed by Lin and He for JST [25]. However, as opposed to having a topic distribution for each sentiment, our model has a sentiment distribution for each topic. As we will explain in Chapter 3, this is a more appropriate approach for the task we are trying to achieve. To determine the overall rating, we make the simplifying assumption that the more a person talks about a topic the more important that topic is to that person. Under that assumption, a topic distribution is proportional to topic weights as defined by Wang et al. in their work on LRR [45] and LARAM [46] so the overall rating can be defined in the same way. Finally, we add a few heuristics for better sentiment target detection and sentiment approximation for aspects that were not mentioned.

## 1.4 Contributions

Our contributions in this thesis are:

1. A New Model for Aspect-Level Sentiment Analysis

   We present SentPOSLDA, a new model that discovers aspects and sentiments expressed towards those aspects in opinionated text as well as capturing semantic and syntactic properties of the words in the text.

2. A Parallel Processing Method for Gibbs Sampling

   Running a Gibbs sampling process for a sufficient number of iterations for SentPOSLDA, and even for some of the previous models, can take a very long time. In this thesis, we present a parallel-processing approach to Gibbs sampling and show that the efficiency of the process improves drastically without compromising the quality of the results.

3. Comparison to Existing Methods

   We present a qualitative and quantitative comparison of our work to the existing approaches and show that our method matches or exceeds the performance of the existing approaches.

## 1.5 Overview

The remainder of this thesis is organized as follows. Chapter 2 provides a review of the concepts and literature relevant to our work, including topic modelling, part-of-speech tagging and sentiment analysis. Chapter 3 presents the model itself, including the generative process and the inference algorithms as well as the parallel-processing approach to these algorithms. Chapter 4 describes the details of our experiments and the subsequent analysis. Chapter 5 summarizes the conclusions and suggests directions for future work.

# Chapter 2

# Background and Related Work

In this chapter we provide the background knowledge necessary for understanding our work. We begin by explaining some general concepts from probability theory. We then show how these concepts have been applied to topic modelling, part-of-speech tagging and sentiment analysis in the recent research. Notable models that have influenced our work are presented in more detail and the reader is directed to the original papers for the details of other approaches.

## 2.1 General Concepts

The following is a brief overview of concepts from Bayesian probability and the related methods and extensions. It is assumed that the reader is familiar with the basics of probability theory such as the definition of probability, joint, marginal and conditional probabilities, and the sum and product rules.

### 2.1.1 Bayesian Probability Theory

Bayesian probability theory [33] allows us to model uncertainty about future events. Unlike the frequentist interpretation of probability theory that relies on repeatable stochastic events, the Bayesian approach defines probability as a degree of belief. We start with some initial belief about the system and update that belief as new data is observed. The foundation of Bayesian probability is Baye's theorem:

$$p(A|B)=\frac{p(B|A)\,p(A)}{p(B)} \tag{2.1}$$

where A and B are events and B is such that p(B) > 0.

Suppose that we want to know if it is going to rain tomorrow. A quick search online tells us that historically, tomorrow's date has experienced rain 30% of the time. A frequentist approach would rely on this figure alone and leave the probability at 0.3. The Bayesian approach allows us to incorporate a prior bias based on other knowledge or intuition. Let's assume that our favourite weatherman has predicted rain. Let's also assume that we have his track record on hand and when it does rain, he predicts it correctly 90% of the time. When it doesn't rain, he makes an incorrect prediction of rain 10% of the time. Let's define event A as “it rains”, $\overline{A}$ as “it doesn't rain” and B as “the weatherman has predicted rain”. We can then apply Baye's theorem to calculate the probability of rain as follows:

$$\begin{aligned} p(A|B) &= \frac{p(B|A)p(A)}{p(B|A)\,p(A)+p(B|\overline{A})\,p(\overline{A})} \\ &= \frac{(0.9)(0.3)}{(0.9)(0.3)+(0.1)(0.7)} \\ &= 0.794 \end{aligned} \tag{2.2}$$

Tomorrow, when we actually observe the weather, we will know if the weatherman's prediction was correct and we will update his track record accordingly. The next time we want to know if it is going to rain we will use the updated values for p(B|A) and p(B|$\overline{A}$) based on the updated track record.

More formally, consider a probabilistic model with parameters Θ and observed data x. The probability of Θ, p(Θ) describes our beliefs before any data is observed and is called the *prior.* The probability of seeing data x given the model, p(x|Θ) is called the *likelihood*. Using Bayes' theorem we can define the *posterior* distribution:

$$p(\Theta|x)=\frac{p(x|\Theta)\,p(\Theta)}{p(x)} \tag{2.3}$$

giving us the uncertainty in our model after observing some data. If the posterior has the same form as the prior then the prior is the *conjugate prior* for the likelihood.

Since the denominator only serves as a normalization constant, Bayes' theorem can be simplified to:

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

The parameters Θ can themselves be defined as probability distributions. In that case, the parameters of the parameters are called *hyperparameters*.

The models described in this thesis are more complex than the weather example, often having the posterior of one variable act as the prior for another. In the next subsection, we will present a tool that can aid in understanding complex models.

### 2.1.2 Bayesian Networks

A Bayesian network is a graphical representation of a joint distribution. Graphical representations are a convenient way to visualize the structure of a model and can be used to obtain insight into its properties by visual inspection.

Each variable in the distribution is represented by a node in the network. A dependence relationship between two nodes is represented by a directed arc to the dependent node from the node it depends on. For example, the Bayesian network representation of the joint distribution p(W, X, Y, Z) = p(W)p(X)p(Y|W)p(Z|X, Y), where W and Z are conditionally independent, is shown in Figure 2.1.

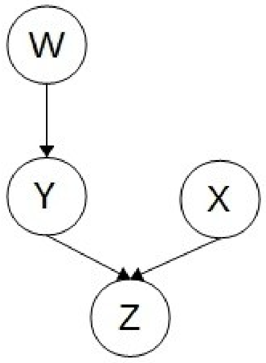


**Figure 2.1:** Graphical representation of a joint probability

If the value of a variable is observable then the node representing that variable is shaded while other, latent variables, are left transparent. For example, when analyzing text, the words are values that are observed while their properties such as topic, part-of-speech, etc. are typically latent. If we wanted to depict a very simple model where variables $w_1$, ..., $w_N$ represent words in a piece of text and each word is dependent on a variable T representing the topic, the resulting Bayesian network would be as in Figure 2.2.

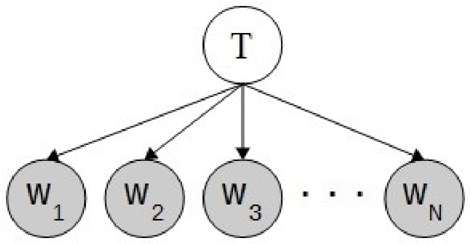


**Figure 2.2:** Observed variables are shaded while latent variables are transparent.

The previous example was of a very simple model and yet the diagram was already somewhat cumbersome due to multiple occurrences of the same type of variable. If we were to include document instances and other repetitive variables into the model, the diagram would become extremely elaborate and barely usable. Natural language processing (NLP) researchers have developed a more compact notation for these purposes called the plate notation. When using the plate notation, one node is drawn with a common label representing the type of variable. The node is placed within a plate with the number of occurrences of that type of variable placed in one of the corners. For example, the model in Figure 2.2. can be drawn using the plate notation as in Figure 2.3.

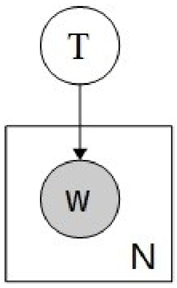


**Figure 2.3:** Plate notation

If we wanted to model multiple documents where each document has its own topic, we simply place the above diagram into another plate with the number of documents in one of the corners:

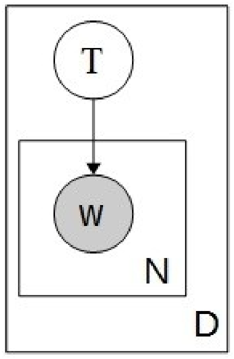


**Figure 2.4:** Modelling document instances is a simple matter of adding another plate.

A common practise in NLP circles that also deviates from tradition is including distribution parameters into the diagram. For example, if we wanted to model each word as being drawn from a multinomial distribution parameterized by θ, where each document has its own value for θ then we would draw our diagram as follows:

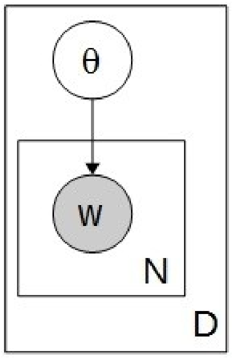


**Figure 2.5:** Distribution parameters are included in NLP diagrams.

The type of distribution is not depicted in the diagram, only the parameters. To fully define a model we describe its generative process. The generative process formally defines the assumed steps taken to generate the observed data. For example, the generative process for the model in the previous example is as follows:

```
1. For each document d:
    (a) For each word $w_i$ in d:
        i. Draw $w_i \sim \text{Multinomial}(\theta_d)$
```

## 2.1.3 Markov Chains and Markov Models

A subclass of Bayesian networks called dynamic Bayesian networks is used to model time series data. Each variable in a dynamic Bayesian network is assigned a time index, for example $\{X_1, \ldots X_T\}$. In one of the simpler cases each variable is only dependent on the previous variable so that the joint distribution of the series is:

$$p(X_1, \ldots, X_T) = p(X_1)\, p(X_2|X_1) \ldots p(X_T|X_{T-1}) \tag{2.4}$$

A system with a probability distribution such as the one above is called a *Markov Chain* and a model based on a Markov Chain is called a *Markov Model*. Figure 2.6 shows a graphical representation.

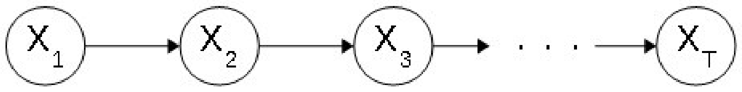


**Figure 2.6:** Markov Chain

The type of model we just described is a 1st-order Markov model but it is also possible to have an nth-order Markov model where each variable is dependent on the previous n variables.

A simple example of a Markov chain is any type of board game where the moves on the board are determined by a roll of a dice. In this scenario the next position on the board is determined only by the current position and the roll of the dice. The events that led to us occupying the current position have no effect on what the next position will be.

Consider now the following scenario. Suppose that our neighbour goes to one of two grocery stores every day. The store he goes to on a given day depends on what he needs that day. He is unlikely to need something from the store he went to the day before since he usually buys enough to last him a few days. However, if he is in the mood for some particular type of food he might go to the same store two days in a row. Since the next store he goes to depends on the last one he went to and on his mood that day, the sequence of stores that he goes to is a Markov chain. Now let's assume that we do not have a view of the path he takes to either store so we can never see which one he went to. Let's also assume that we have a dog that runs out through a doggy door to greet our neighbour every time he comes back from the store. Our neighbour happens to like dogs so every time he goes to a store he buys a treat for our dog if he has enough money left over. The treats are more expensive at one store than the other so he is less likely to have enough money left for a treat when he shops at the more expensive store. Every day, we see our dog leaving and entering the house and we observe whether or not the dog came back with a treat in his mouth. From those observations and knowing the rough pattern our neighbour typically follows we can deduce which store he likely went to on a given day.

The scenario we just described is an example of a *Hidden Markov Model* (HMM). Formally, if $\{X_1, ..., X_T\}$ is a Markov chain such that $X_1,... X_T$ are discrete variables that we cannot observe and $Y_1, ..., Y_T$ are variables that we can observe such that each $Y_t$ is dependent on $X_t$ for $1 < t < T$ then the resulting model is an HMM. Figure 2.7 shows a graphical representation.

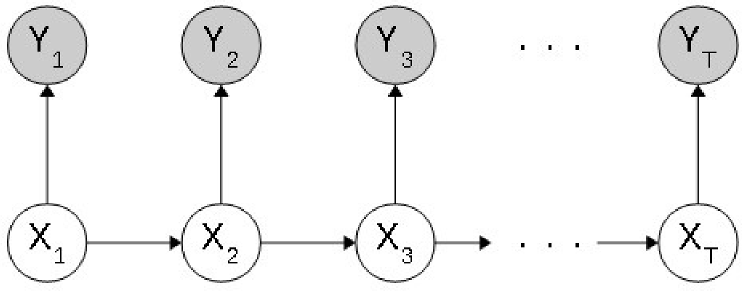


**Figure 2.7:** Hidden Markov Model

The hidden variables are also called state variables. We are typically interested in deriving the probability of the hidden variables based on the observations of the visible variables. To that end, we use the joint probability distribution of an HMM:

$$p(X_1,\dots,X_T,Y_1,\dots,Y_T)=p(X_1)p(Y_1|X_1)\prod_{t=2}^{T}p(X_t|X_{t-1})p(Y_t|X_t) \tag{2.5}$$

If the $X_t$ variables have K possible values then the $p(X_t|X_{t-1})$ distribution can be defined by a K × K state *transmission matrix*. Similarly, if $Y_t$ variables are discrete and have L possible values then the $p(Y_t|X_t)$ distribution can be defined by a K × L *emission matrix*.

As we shall see later in this thesis, HMMs have been very useful in modelling syntax in text and we will employ them in our model for the same purpose.

### 2.1.4 Approximate Inference with Gibbs Sampling

When applying Bayesian probability to natural language processing we will typically define a probabilistic generative process that models the way a person chooses their words based on latent properties such as topic, sentiment, syntax, etc. We will then try to invert that process using statistical inference to calculate distributions over those latent properties conditioned on a data set. However, doing such a calculation is often intractable. In these cases we have several options for approximate inference. The approach we have taken in this thesis is a form of Markov Chain Monte Carlo algorithms called Gibbs sampling.

Monte Carlo techniques are algorithms for obtaining the desired value through simulations based on probabilistic choices. As a very simple example, we can calculate the approximate value of the constant π with the following Monte Carlo method. Draw a large square on the ground of size d x d and within it draw a circle of diameter d (ie. the circle just barely fits inside the square). Now sprinkle some rice uniformly over the drawing. The ratio of the number of grains of rice inside the square, S and the number of grains of rice inside the circle, C should approximate the ratio between the area of the square and the area of the circle. Therefore,

$$\frac{C}{S} \approx \frac{\pi (d/2)^2}{d^2} \tag{2.6}$$

If we observed that C = 100 and S = 127, solving for π would get us:

$$\pi \approx 4\frac{C}{S} = 4\frac{100}{127} = 3.149606299 \tag{2.7}$$

If we were to throw some more grains of rice on the drawing we would have more data points and our approximation would become more accurate. As the number of grains approaches infinity the approximation approaches the exact value.

In the example above we used a sampling technique to approximate a property of a simple model involving a uniform distribution. Going back to performing inference on complex probability distributions, let $X = \{x_1, ..., x_N\}$ represent observed data and $Y = \{y_1, ..., y_N\}$ some latent properties of that data. For example, X can be the words in a corpus of documents and Y can be topics associated with those words. Given the probability p(X, Y), we want to derive an approximation to p(Y|X). There are several sampling techniques that can be used to do this but the main idea to keep in mind is that the samples can be seen as a sequence of transitions through a state space (ie. $Y^{(1)}$ to $Y^{(2)}$ to $Y^{(3)}$, etc). From that perspective, the process can be expressed as:

1. $Y^{(0)}$ = random point

2. For t = 1 to T:

    (a) Draw $Y^{(t)} \sim p(Y^{(t)} \mid Y^{(0)}, ..., Y^{(t-1)}, X)$

Where T is the total number of samples we want to take. Our goal is to perform these transitions in a way that is proportional to $p(Y|X)$. In other words, we want the transitions to lead us to the states where $p(Y|X)$ is high more often than to the states where it is low. It can be shown (ex. [4]) that if we setup $p(Y^{(t)} \mid Y^{(0)}, ..., Y^{(t-1)}, X)$ as a Markov Chain (this is the "Markov Chain" part of "Monte Carlo Markov Chain") so that $p(Y^{(t)} \mid Y^{(0)}, ..., Y^{(t-1)}, X) = p(Y^{(t)} \mid Y^{(t-1)}, X)$ and if that chain satisfies certain conditions then the stationary distribution of that chain will indeed be proportional to the $p(Y|X)$ distribution. The Gibbs sampling algorithm was designed to meet precisely those conditions [4].

After a random initialization of the variables, the Gibbs sampler draws a value for each of the variables from its probability distribution conditioned on the values of all the other variables. The algorithm repeats the drawing step until the values converge, although typically we set some iteration limit instead of testing for convergence at each iteration.

More formally, the Gibbs sampling algorithm performs the following steps:

```
1. Randomly initialize $y_1^{(0)}, ..., y_N^{(0)}$
2. For t = 1, ..., T:
        (a) Draw $y_1^{(t+1)} \sim p(y_1|y_2^{(t)}, ..., y_N^{(t)}, x_1)$
        (b) Draw $y_2^{(t+1)} \sim p(y_2|y_1^{(t+1)}, y_3^{(t)}, ..., y_N^{(t)}, x_2)$
        ....
        (c) Draw $y_N^{(t+1)} \sim p(y_N|y_1^{(t+1)}, ..., y_{N-1}^{(t+1)}, x_N)$
```

Note that the Gibbs sampler does not need to sample from the exact form of the distribution but can instead use a simpler function that is proportional to the distribution. This can make the sampling step simpler and more efficient. For simplicity and efficiency, the probabilities are often (and for all the models in this thesis) expressed in terms of counts of the variable values and value co-occurrences. When the probability functions take such a form the algorithm becomes:

```
1. Randomly initialize $y_1^{(0)}, ..., y_N^{(0)}$
2. For t = 1, ..., T:
   (a) For i = 1, ..., N:
       i.   Decrement all counts involving $y_i$
       ii.  Calculate $p(y_i \mid y_1, ..., y_{i-1}, y_{i+1},..., y_N, x_i)$
       iii. Draw $y_i \sim p(y_i \mid y_1, ..., y_{i-1}, y_{i+1},..., y_N, x_i)$
       iv.  Increment counts to include $y_i$
```

The fact that we can use this form of the algorithm will be important later when we present our parallel-processing implementation.

When the probability distribution contains hyperparameters, they are typically updated at regular iteration intervals. With most implementations the updates do not begin until a certain “burn in” period has passed since the sampled values during the early iterations are still far off from the original distribution.

## 2.2 Topic Modelling

### 2.2.1 Early Approaches

Although not a probabilistic model with a Bayesian formulation, the root of modern topic models is a matrix factorization approach called Latent Semantic Indexing (LSI) [11]. In LSI, topics are modelled with linearly independent base vectors and topic classification is achieved by decomposing the matrix representation of the corpus into such vectors.

Inspired by LSI, but striving for a generative probabilistic model, Hoffman [19] developed Probabilistic Latent Semantic Analysis (pLSA). In pLSA, each word is described by a mixture model where the components of that model are multinomial random variables representing topics. Specifically, it assumes the following generative process:

1. For each word $w_n$ in document d:

   (a) Draw a topic $z_n \sim \text{Multinomial}(\theta_d)$

   (b) Draw a word $w_n \sim \text{Multinomial}(\phi_{z_n})$

where $\theta_d = p(z|d)$ and $\phi_z = p(w|z)$. A graphical representation is shown in Figure 2.8 where D is the number of documents and N is the total number of words.

The joint probability of word $w_n$ and document d is then:

$$p(w_n, d) = p(d) \sum_z p(w_n|z)\, p(z|d) \tag{2.8}$$

The generative process can then be reversed through posterior inference to determine the topic assignments $z_n$.

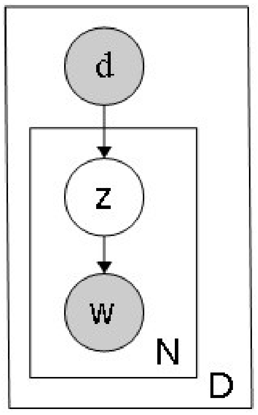


**Figure 2.8:** pLSA model

The problem with this approach is that there is no probabilistic model for the proportion of topics in a given document. The d in the above equation is simply an index in the list of documents in the training set. That means that the number of parameters grows linearly with the size of the corpus and that leads to problems with overfitting. It also makes it impossible to perform inference for documents outside the training set. Latent Dirichlet Allocation [7], described in the next section, solves this problem by giving the topic distribution a Dirichlet prior.

### 2.2.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) [7] is a generative probabilistic model for collections of discrete data. Although it is general enough for modelling any kind of data, it was originally introduced as a topic model and has been a popular basis for more complex text modelling problems (ex. [6], [5], [36], [44], [38]). In the context of text modelling, the documents are defined as random mixtures over latent topics and each topic is represented as a distribution over words. More specifically, LDA assumes the following generative process:

1. For each topic $z$:

    (a) Draw topic-word distribution $\phi_t \sim$ Dirichlet($\beta$)

2. For each document $d$:

    (a) Draw document-topic distribution $\theta_d \sim$ Dirichlet($\alpha$)

    (b) For each word $w_i$ in $d$:

        i. Draw a topic $z_i \sim$ Multinomial($\theta_d$)

        ii. Draw a word $w_i$ from $p(w_i \mid z_i, \phi^{(t_i)})$

Where $\alpha$ and $\beta$ are the model hyperparameters for the Dirichlet priors. A graphical representation of the model is given in Figure 2.9. The cardinality T in the diagram is the number of topics.

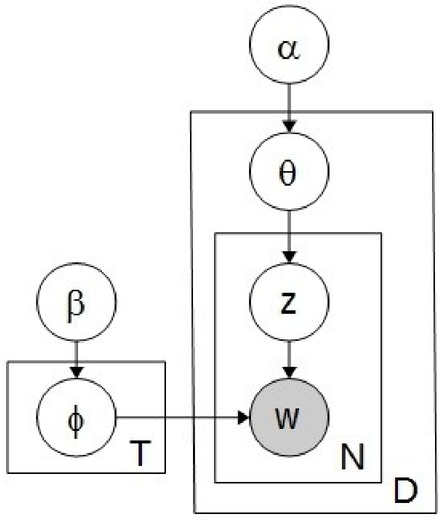


**Figure 2.9:** Latent Dirichlet Allocation model

We stayed at the Loyal Inn because we wanted to be in downtown Seattle, since we only had two nights there. It exceeded our expectations in many respects. The rooms were large and comfortable, with many extra amenities. The breakfast was very extensive, with many fresh fruits, eggs, breads and even waffles. The hotel staff was very friendly and helpful, especially with suggestions of places to see during our rather limited time in the city. (Highlights were the Columbia Center, the second tallest building on the west coast, with marvellous views of the surrounding area, and also the Pike Street Market.)We also appreciated the location, as it is within two blocks of the area of free bus transportation from early morning to late evening on all buses in the downtown area. This was a marvellous idea, and did much to cut down on traffic congestion. I plan to return to this hotel in the spring of 2009.

| | | | |
|---|---|---|---|
| Value | Location | Front Desk | Business Service |
| Room | Cleanliness | Service | Other |

**Figure 2.10:** Sample review with topics detected by LDA. (Topic labels added manually.)

Once the distributions for a given corpus have been learned, each document d will be characterized by its document-topic distribution $\theta_d$. For example, suppose that our corpus contains the review in Figure 2.10 and LDA, with T=8, detects the topics as indicated by the colour scheme. About 10% of the detected topics accounted for the room, 10% for the service, 10% for the front desk, 20% for the location and the remaining 50% for other topics. Value, cleanliness, and business service were not detected at all. If we index the topics in the order they appear in the legend then the document-topic distribution for this particular document will be $\theta_d = (0, 0.1, 0.2, 0, 0.1, 0.1, 0, 0.5)$.

The full joint probability for LDA is given by:

$$p(w, z, \theta, \varphi | \alpha, \beta) = \prod_{t=1}^{T} Dir(\varphi_t | \beta) \prod_{d=1}^{D} Dir(\theta_d | \alpha) \prod_{d=1}^{D} \prod_{i=1}^{N_d} p(z_{d,i} | \theta_d) \prod_{d=1}^{D} \prod_{i=1}^{N_d} p(w_{d,i} | \varphi_{z_{d,i}}) \qquad (2.9)$$

Exact inference in this case is, of course, intractable and necessitates an approximate inference technique. Blei, et al., the original authors of LDA, used a technique called variational inference to calculate the p(z|w) probability distribution. Since that technique is not relevant to our work we will refer the reader to the original LDA paper for further details and present an alternative approach using Gibbs sampling.

In the notation we used in section 2.4.1, Y = z and X = w. The entire derivation of a Gibbs sampler for LDA is rather lengthy and can be found in [43] and [8]. Here, we simply present the resulting, simplified probability function for the sampler:

$$p(z_i|z_{-i}, w_i) \propto \left\{ \frac{n_{w_i}^{(z_i)} + \beta}{n^{(z_i)} + W\beta} \cdot \frac{n_{z_i}^{(d)} + \alpha_{z_i}}{n^{(d)} + \alpha} \right. \tag{2.10}$$

where $w_i$ is the observed word, $z_i$ is the topic assigned to $w_i$, $z_{-i}$ are the topic assignments of all words except $w_i$, $n_{z_i}^{(d)}$ is the number of times topic $z_i$ occurs in document d, $n^{(d)}$ is the number of times any topic was mentioned in document d, $n_{w_i}^{(z)}$ is the number of times the word $w_i$ was used for topic $z_i$ and $n^{(z_i)}$ is the number of times topic $z_i$ was mentioned in the corpus. This function can then be used in the Gibbs sampling algorithm described in section 2.4.1 to get the desired p(z|w) distribution.

Blei, et al. [7] reported qualitative results by showing a sample grouping of words into topics. Table 2.1 shows some sample topics learned from the TREC AP corpus. They also reported quantitative results in terms of perplexity of a held-out test set. Perplexity is a common way to measure performance in language modelling and can be described as the average predicted number of equally likely words for a given position. More formally, for a held-out test set of size M, $D_{test} = (\mathbf{w}_d)^M_{d=1}$, perplexity is defined as:

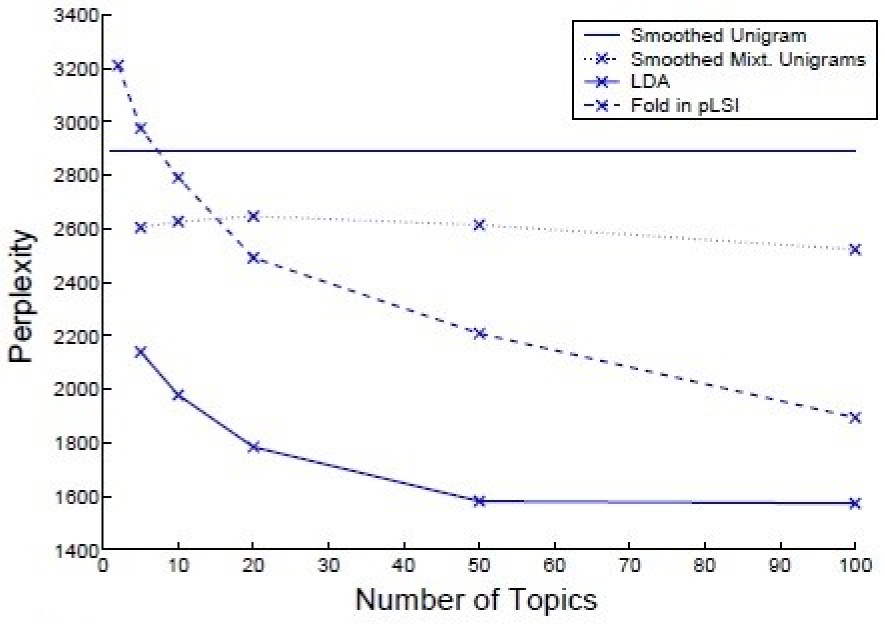
$$perplexity(D_{test}) = \exp\left( - \frac{\sum_{d=1}^{M} \log p(\boldsymbol{w}_d|\Theta)}{\sum_{d=1}^{M} N_d} \right) \tag{2.11}$$

where Θ, the model parameters, are learned from the training data, $p(\mathbf{w}_d|\Theta)$ is the likelihood of document $\mathbf{w}_d$ given those parameters and $N_d$ is the number of words in document $\mathbf{w}_d$. Performance is inversely proportional to the perplexity score. Figure 2.11 shows the perplexity graphs as compared to other models on two data sets.

The rest of the models described in this chapter that include a topic classification component all have LDA as their basis. Our own model follows that approach as well.

| Arts | Budgets | Children | Education |
|---|---|---|---|
| new | million | children | school |
| film | tax | women | students |
| show | program | people | schools |
| music | budget | child | education |
| movie | billion | years | teachers |
| play | federal | families | high |
| musical | year | work | public |
| best | spending | parents | teacher |
| actor | new | says | Bennett |
| first | state | family | manigat |
| york | plan | welfare | namphy |
| opera | money | men | state |
| theater | programs | percent | president |
| acress | government | care | elementary |
| love | congress | life | diti |

**Table 2.1:** Example topics learned from TREC AP corpus with LDA



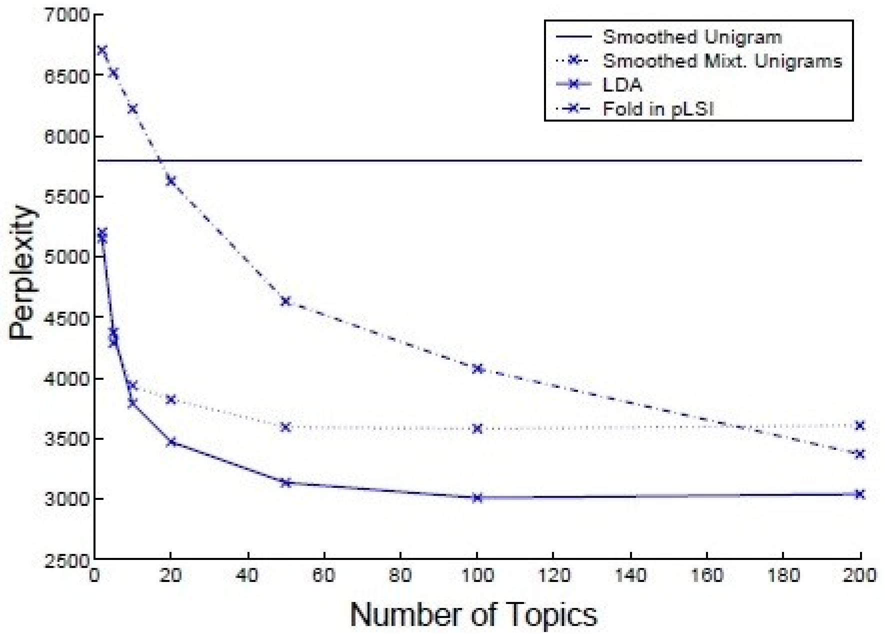


**Figure 2.11:** Perplexity results on the nematode (Top) and AP (Bottom) corpora for LDA, the unigram model, mixture of unigrams, and pLSI

## 2.3 Syntax Modelling

Early attempts to model syntax relied on extensive rules about the grammatical structure of text. Accounting for every possible scenario and keeping up with the changing nature of language, however, proved to be exceedingly difficult. In one case, Green and Rubin [15] created a POS tagger that involved several thousand rules and still achieved accuracy of only 77%.

Natural language seems to be very flexible and varied and is therefore an excellent candidate for statistical or probabilistic approaches. When analyzing a piece of text, the words are observable to us and we know that each word was chosen partially due to the POS class that is likely to follow the POS class of the word before it. It is therefore natural to model this problem with an HMM; the POS classes are the hidden states and the words are the observable variables generated from those states.

In the models described in this thesis and in our own model, syntax is modelled using a Bayesian HMM. In this type of HMM the transition rows and emission probabilities are multinomial random variables with Dirichlet priors. Figure 2.12 shows a graphical representation. In the diagram, $c_i$ is the POS class assignment for word $w_i$ and C is the number of POS classes.

The Bayesian HMM approach allows us to use the same framework for Gibbs sampling as for LDA. For a multinomial distribution $p(x|\theta)$ with a Dirichlet prior $p(\theta|\alpha)$, the probability that $x_i$ is assigned $k \in K$ is:

$$p(x_i = k | x_{-i}, \alpha) = \frac{n_k + \alpha}{n + |K|\alpha - 1} \tag{2.12}$$

Where $\alpha$ is a symmetric K-vector and $x_{-i}$ are the assignments of all variables except $x_i$.

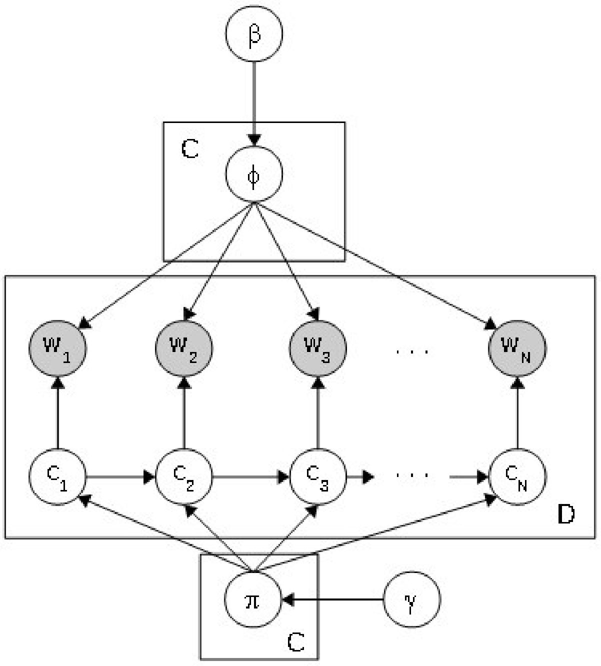


**Figure 2.12:** Bayesian HMM for syntax modelling

Therefore, the probability of part of speech class $c_i$ given $c_{-i}$ in the Bayesian HMM is:

$$p(c_i|c_{-1},\gamma) \propto \frac{n_{c_{i-2},c_{i-1},c_i}+\gamma_{c_i}}{n_{c_{i-2},c_{i-1}}+\gamma} \tag{2.13}$$

where $n_x$ is the number of times sequence x of POS classes appears in the corpus. The probability of $c_i$ given $c_{-i}$ and words w is:

$$p(c_i|c_{-i},w,\beta,\gamma) \propto \frac{n_{w_i}^{(c_i)}+\beta}{n^{(c_i)}+W\beta}\,\frac{n_{(c_{i-2},c_{i-1},c_i)}+\gamma_{c_i}}{n_{(c_{i-2},c_{i-1})}+\gamma_.}\,\frac{n_{(c_{i-1},c_i,c_{i+1})}+\gamma_{c_i}}{n_{(c_{i-1},c_i)}+\gamma_.}\,\frac{n_{(c_i,c_{i+1},c_{i+2})}+\gamma_{c_i}}{n_{(c_i,c_{i+1})}+\gamma_.} \tag{2.14}$$

where $n_{w_i}^{(c_i)}$ is the number of times that $c_i$ is the POS class of word $w_i$ and $n^{(c_i)}$ is the total number of occurrences of class $c_i$.

## 2.4 Integrating Topics and Syntax

In [16], Griffits et al. combine the LDA and Bayesian HMM models to model topics and syntax simultaneously. In their model, all semantic words (ie. words that express a topic) are grouped into a single state in the Bayesian HMM while other, purely syntactic words, are modelled by the remaining states representing different POS classes. Within the semantic state the individual topics are modelled using LDA. A graphical representation is shown in Figure 2.13. We use $C_{SYN}$ to denote the number of purely syntactic words and assume that state 1 is the semantic state.

The formal generative process of HMMLDA is as follows:

```
1. Draw $\theta^{(d)} \sim \text{Dirichlet}(\alpha)$
2. For each word $w_i$ in document d
    (a) Draw topic $z_i \sim \text{Multinomial}(\theta^{(d)})$
    (b) Draw POS class $c_i$ from $\pi^{(c_{i-1})}$
    (c) If $c_i = 1$:
        i. Draw word $w_i \sim \phi(z_i)$
    (d) Else:
        i. Draw word $w_i \sim \phi(c_i)$
```

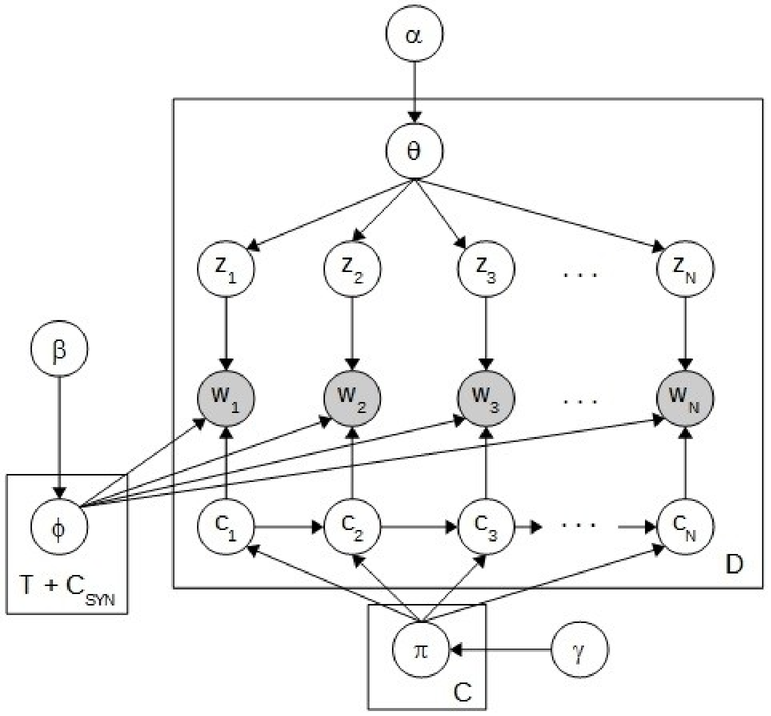


**Figure 2.13:** HMMLDA model

The probability function for the Gibbs sampler is then a combination of equations (2.10) and (2.14):

$$p(c_i, z_i | c_{-i}, z_{-i}, w) \propto \begin{cases} p_{c_i} \cdot \dfrac{n_{w_i}^{(c_i, z_i)} + \beta}{n^{(c_i, z_i)} + W\beta} \cdot \dfrac{n_{z_i}^{(d)} + \alpha_{z_i}}{n^{(d)} + \alpha} & c_i = 1 \\ p_{c_i} \cdot \dfrac{n_{w_i}^{(c_i)} + \beta}{n^{(c_i)} + W\beta} & c_i \neq 1 \end{cases} \tag{2.15}$$

where

$$p_{c_i} = \frac{n_{(c_{i-2}, c_{i-1}, c_i)} + \gamma_{c_i}}{n_{(c_{i-2}, c_{i-1})} + \gamma} \cdot \frac{n_{(c_{i-1}, c_i, c_{i+1})} + \gamma_{c_i}}{n_{(c_{i-1}, c_i)} + \gamma} \cdot \frac{n_{(c_i, c_{i+1}, c_{i+2})} + \gamma_{c_i}}{n_{(c_i, c_{i+1})} + \gamma} \tag{2.16}$$

Figure 2.14, taken from [16], shows some sample experiment results on a concatenated Brown and TASA corpora in comparison to LDA.

(a) LDA topics

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| the | the | the | the | the | a | the | the | the |
| blood | , | , | of | a | the | , | , | , |
| , | and | and | , | of | of | of | a | a |
| of | of | of | to | , | , | a | of | in |
| body | a | in | in | in | in | and | and | game |
| heart | in | land | and | to | water | in | drink | ball |
| and | trees | to | classes | picture | is | story | alcohol | and |
| in | tree | farmers | government | film | and | is | to | team |
| to | with | for | a | image | matter | to | bottle | to |
| is | on | farm | state | lens | are | as | in | play |

(b) HMMLDA topics

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| blood | forest | farmers | government | light | water | story | drugs | ball |
| heart | trees | land | state | eye | matter | stories | drug | game |
| pressure | forests | crops | federal | lens | molecules | poem | alcohol | team |
| body | land | farm | public | image | liquid | characters | people | * |
| lungs | soil | food | local | mirror | particles | poetry | drinking | baseball |
| oxygen | areas | people | act | eyes | gas | character | person | players |
| vessels | park | farming | states | glass | solid | author | effects | football |
| arteries | wildlife | wheat | national | object | substance | poems | marijuana | player |
| * | area | farms | laws | objects | temperature | life | body | field |
| breathing | rain | corn | department | lenses | changes | poet | use | basketball |

(c) HMMLDA classes

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| the | in | he | * | be | said | can | time | , |
| a | for | it | new | have | made | would | way | ; |
| his | to | you | other | see | used | will | years | ( |
| this | on | they | first | make | came | could | day | : |
| their | with | i | same | do | went | may | part | ) |
| these | at | she | great | know | found | had | number | |
| your | by | we | good | get | called | must | kind | |
| her | from | there | small | go | | do | place | |
| my | as | this | little | take | | have | | |
| some | into | who | old | find | | did | | |

**Figure 2.14:** (a) Topics extracted by the LDA model. (b) Topics from the composite model. (c) Classes from the composite model.

As we can see, the HMMLDA model performs a cleaner separation of semantic and syntactic words and the semantic words are more descriptive of the detected topics.

In [9], Darling extends HMMLDA to model the different types of semantic POS classes. In POSLDA, instead of grouping all semantic words into a single state, the states are split into semantic and syntactic (non-semantic) states. Semantic words are then modelled using a joint probability distribution of a Bayesian HMM and LDA.

If we denote the number of semantic classes as $C_{SEM}$ and we choose the first $C_{SEM}$ states to be the semantic states, then the formal generative process for POSLDA is as follows:

```
1. For each row $\pi_r$ in $\pi$:
    (a) Draw $\pi_r \sim$ Dirichlet($\gamma$)
2. For each word distribution $\phi_t \in \phi$:
    (a) Draw $\phi_t \sim$ Dirichlet($\beta$)
3. For each document d:
    (a) Draw topic distribution $\theta_d \sim$ Dirichlet($\alpha$)
    (b) For each word token $w_i$ in document d:
        i. Draw POS clas $c_i \sim \pi_{ci-1}$
        ii. If $c_i \leq C_{SEM}$:
            A. Draw word $w_i \sim \phi_{ci}^{(SYN)}$
        iii. Else:
            A. Draw topic $z_i \sim \theta_d$
            B. Draw word $w_i \sim \phi_{ci,zi}^{(SEM)}$
```

The probability function for the Gibbs sampler is an extension to the one for HMMLDA:

$$p(c_i, z_i | c_{-i}, z_{-i}, w) \propto \begin{cases} p_{c_i} \cdot \dfrac{n_{w_i}^{(c_i,z_i)} + \beta}{n^{(c_i,z_i)} + W\beta} \cdot \dfrac{n_{z_i}^{(d)} + \alpha_{z_i}}{n^{(d)} + \alpha} & c_i \le C_{SEM} \\ p_{c_i} \cdot \dfrac{n_{w_i}^{(c_i)} + \beta}{n^{(c_i)} + W\beta} & c_i > C_{SEM} \end{cases} \tag{2.17}$$

where $p_{ci}$ is the same as in (2.16). A graphical representation of POSLDA is presented in Figure 2.15.

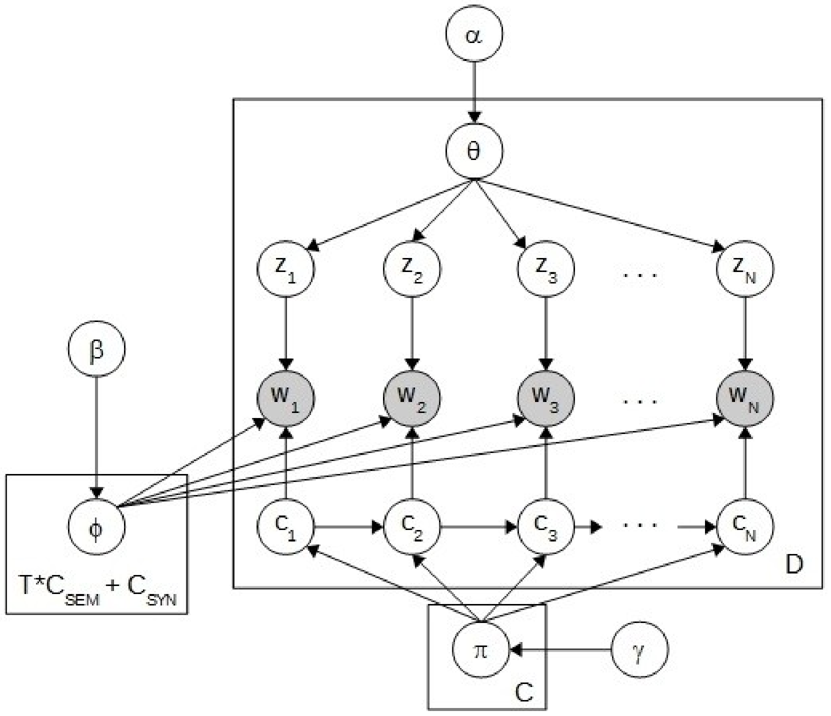


**Figure 2.15:** POSLDA model

Notice that LDA, Bayesian HMM and HMMLDA are special cases of POSLDA; setting C = 1 reduces the model to LDA, setting T = 1 reduces it to the Bayesian HMM and setting $C_{SEM} = 1$ reduces it to HMMLDA. One advantage of such a generalized structure is that experiments involving all four models can be easily performed using a single implementation of POSLDA just by modifying the relevant parameters.

Darling performed experiments on the TREC AP data set comparing POSLDA to both LDA and HMMLDA on the task of topic modelling. POSLDA outperformed both as can be seen in Figure 2.16.

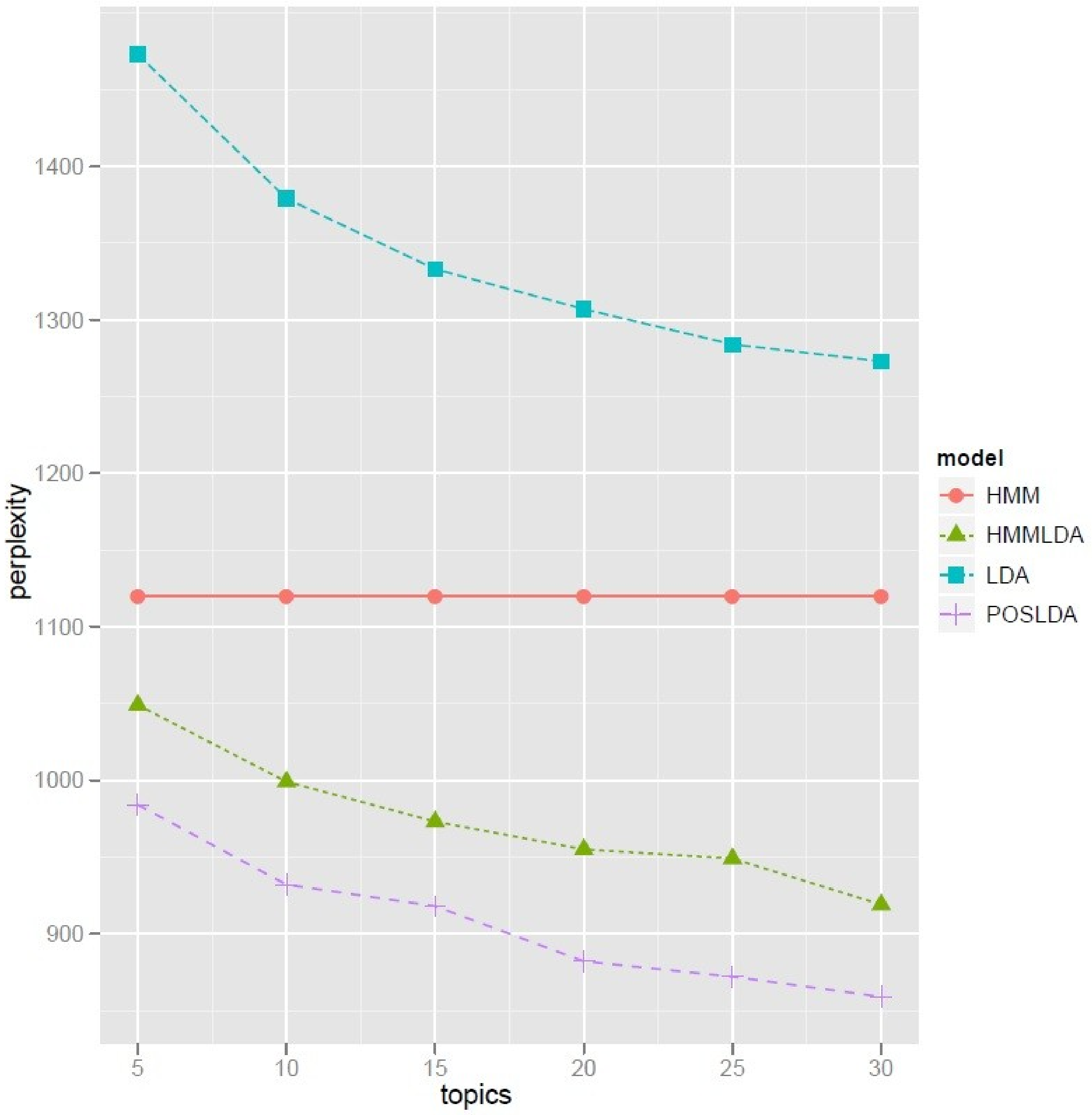


**Figure 2.16:** Perplexity of POSLDA and other probabilistic models as T varies.

## 2.5 Sentiment Analysis

### 2.5.1 Early Approaches

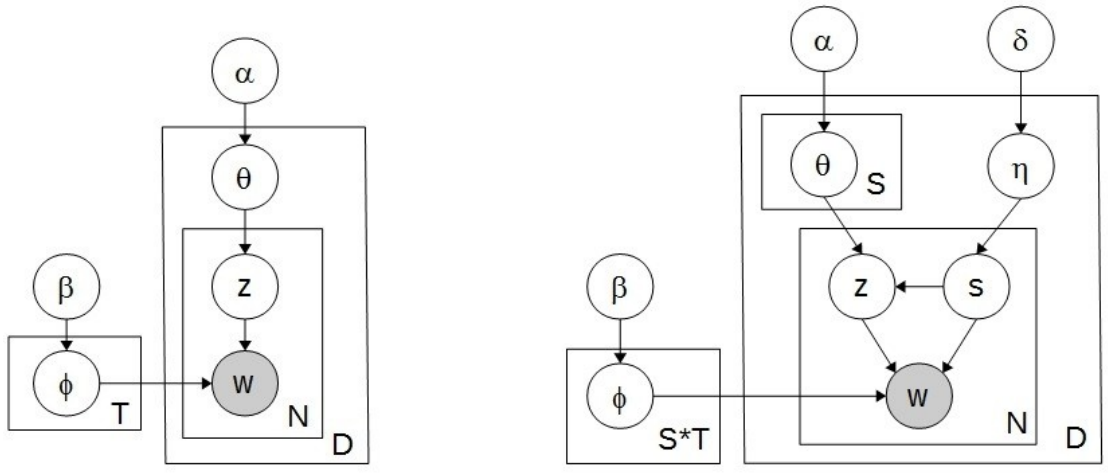
Arguably, the paper that kicked off modern day research into sentiment analysis was the 1997 paper by Hatzivassiloglou and McKeown [17] on the semantic orientation of adjectives. Hatzivassiloglou and McKeown took advantage of conjunctions and disjunctions of adjectives in large corpora to detect sentiment polarities of individual adjectives. Subsequently, in the early 2000s, research activity in this area saw an exponential increase (ex. [18], [47], [48], [39], [40], [41], [32], [10]).

Most of this early work was on classifying entire documents as either positive or negative based on the polarity of words found within them. In the famous 2002 paper [41], Turney used the Pointwise Mutual Information and Information Retrieval (PMI-IR) [40] algorithm to calculate the similarity of phrases containing adjectives and adverbs to the word “excellent” and the word “poor”. He utilized a POS tagger to identify the adjectives and adverbs, a set of hardcoded POS patterns to identify opinionated phrases and a web search engine to retrieve the necessary statistics. A review was classified as positive or negative according to the average semantic orientation of the identified phrases. In his experiments, the accuracy ranged from 66% for movie reviews to 84% for car reviews. Although the implementation was somewhat impractical, the work demonstrated the potential of further research in this area. In [32], Pang et al. performed experiments on the same task using three machine learning algorithms: Naive Bayes, maximum entropy and support vector machines. They found that although these methods showed promising results, they did not perform as well on classifying by sentiment as they did on classifying by topic. They further noted that using only adjectives gave worse performance than using all types of words. In retrospective, these results make sense considering all the challenges mentioned in Chapter 1.

Later on, researchers began work on more complex problems such as sentiment analysis on a multi-point scale (ex. [14], [31]) and the focus of our research, aspect-level sentiment analysis. The initial aspect-level models were largely lexicon-based or utilized a bootstrapped approach (ex. [50], [21], [12], [20], [23], [34]). Such approaches were often too rigid to handle domain dependence and other challenges mentioned in Chapter 1. The more modern approach has been to extend probabilistic topic models such as LDA. We describe some of the more notable work in this area in the next section.

### 2.5.2 Probabilistic Models

To our knowledge, Joint Sentiment/Topic model (JST) [25] was the first fully unsupervised probabilistic model to detect topic and sentiment simultaneously. It extended LDA by adding an additional layer for modelling sentiment. Figure 2.17 shows a graphical representation along with LDA for comparison.



(a) LDA (b) JST

**Figure 2.17:** Graphical representations of LDA and JST

The S in the diagram is the number of sentiment labels, s is the sentiment orientation of a word and $\eta$ is the overall sentiment distribution for a document. The formal generative process assumed by JST is as follows:

1. For each document d:

    (a) Draw sentiment distribution $\eta_d \sim \text{Dirichlet}(\delta)$

    (b) For each sentiment s:

        i. Draw topic-sentiment distribution $\theta_{d,s} \sim \text{Dirichlet}(\alpha)$

2. For each word $w_i$ in document d:

    (a) Draw sentiment $s_i \sim \eta_d$

    (b) Draw topic $z_i \sim \theta_{d,si}$

    (c) Draw word $w_i \sim \phi_{zi,si}$

In [22], Jo and Oh noted that in JST individual words can come from different language models. They argued that if the words from a single sentence came from the same model then each model would focus on the regional co-occurrences of the words in a document. With that in mind, they developed Aspect Sentiment Unification Model (ASUM) with the following generative process:

1. For every pair of sentiment s and topic z:

    (a) Draw $\phi_{s,z} \sim \text{Dirichlet}(\beta_s)$

2. For each document d:

    (a) Draw sentiment distribution $\eta_d \sim \text{Dirichlet}(\delta)$

    (b) For each sentiment s:

i. Draw topic distribution $\theta_{d,s} \sim$ Dirichlet($\alpha$)

(c) For each sentence:

i. Draw sentiment s ~ Multinomial($\eta_d$)

ii. Draw topic z ~ Multinomial($\theta_{d,j}$)

iii. Generate words w ~ Multinomial($\phi_{s,z}$)

Figure 2.18 shows a graphical representation in comparison with JST. The M in the diagram is the number of sentences.

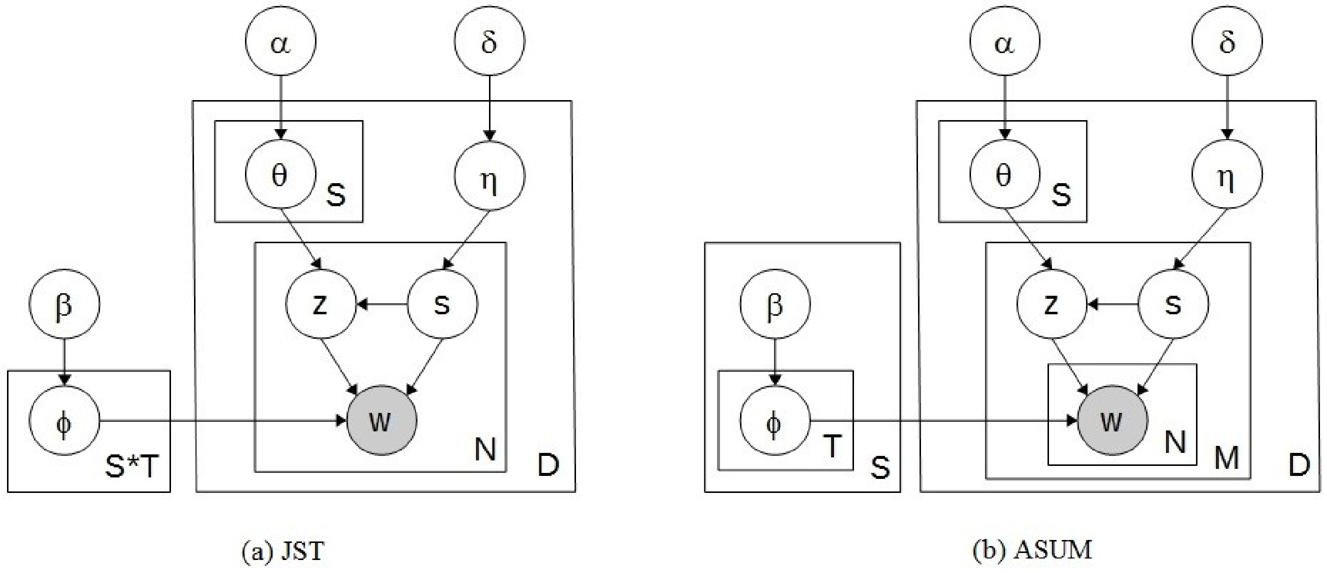


**Figure 2.18:** Graphical representations of JST and ASUM

Li et al. pointed out in [24] that the assumption made in ASUM that every word in a particular sentence expresses the same topic and sentiment was too strong. They further noted that not every word has a sentiment polarity. The model they present, Sentiment Topic Model with Decomposed Prior (STDP), eliminated the sentence-level constraint and introduced a way to model the probability that a word expresses sentiment based on its POS class. However, they do not model POS classes as we do but instead use a POS tagger to provide that information.

Formally, the STDP generative process is as follows:

```
1. For every pair of sentiment s and topic z:
    (a) Draw $\phi_{z,s} \sim \mathrm{Dirichlet}(\beta_{z,s})$
2. For every topic z:
    (a) Draw $\phi_{z,(S+1)} \sim \mathrm{Dirichlet}(\beta_{z,(S+1)})$
3. For each POS group p:
    (a) Draw $\rho_p \sim \mathrm{Beta}(\varepsilon_p)$
4. For each document d:
    (a) Draw sentiment distribution $\eta_d \sim \mathrm{Dirichlet}(\delta)$
    (b) For each $s \in \{0, ..., S+1\}$:
        i. Draw topic distribution $\theta_{d,s} \sim \mathrm{Dirichlet}(\alpha)$
    (c) For each word w in d:
        i. Draw sentiment/non-sentiment indicator $f \sim \mathrm{Bernoulli}(\rho_p)$
        ii. If f = true:
            A. Draw sentiment $s \sim \mathrm{Multinomial}(\eta_d)$
            B. Draw topic  $z \sim \mathrm{Multinomial}(\theta_{d,s})$
            C. Draw word $w \sim \mathrm{Multinomial}(\phi_{z,s})$
```

iii. Else:

A. Draw topic z ~ Multinomial($\theta_{d,(S+1)}$)

B. Draw word w ~ Multinomial($\phi_{z,(S+1)}$)

Figure 2.19 shows the graphical representation of STDP along with ASUM for comparison. Note that the STDP model has S+1 sentiment labels where the extra label models non-sentiment. The P in the model is the number of POS groups.

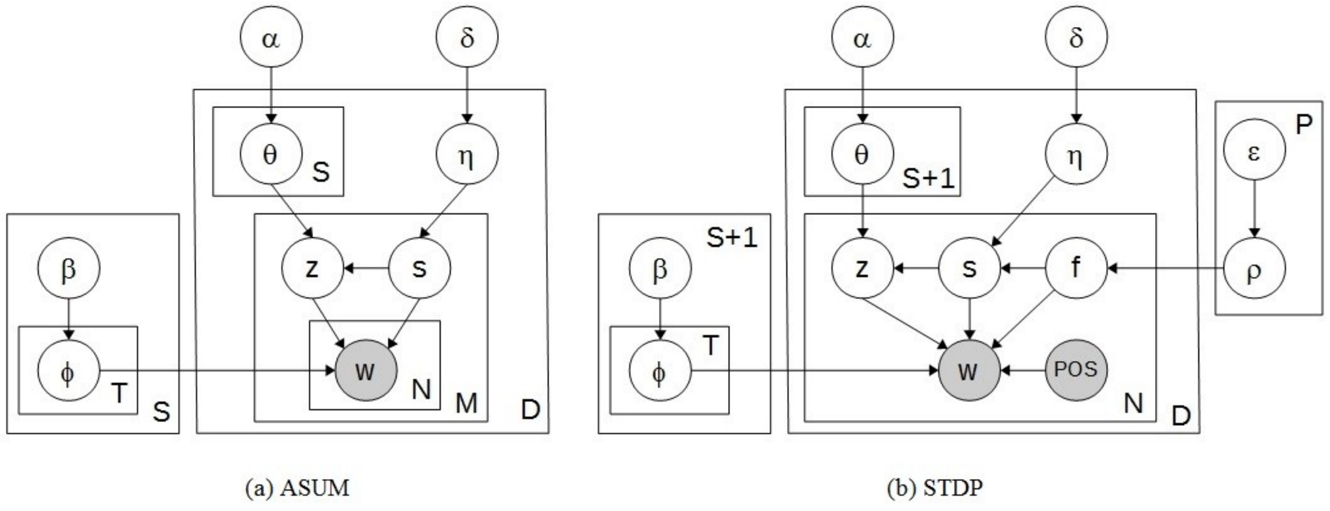


**Figure 2.19:** Graphical representations of ASUM and STDP

The performance comparison between the three models is best summarized by the experiment results in [24] and is presented in Figure 2.20.

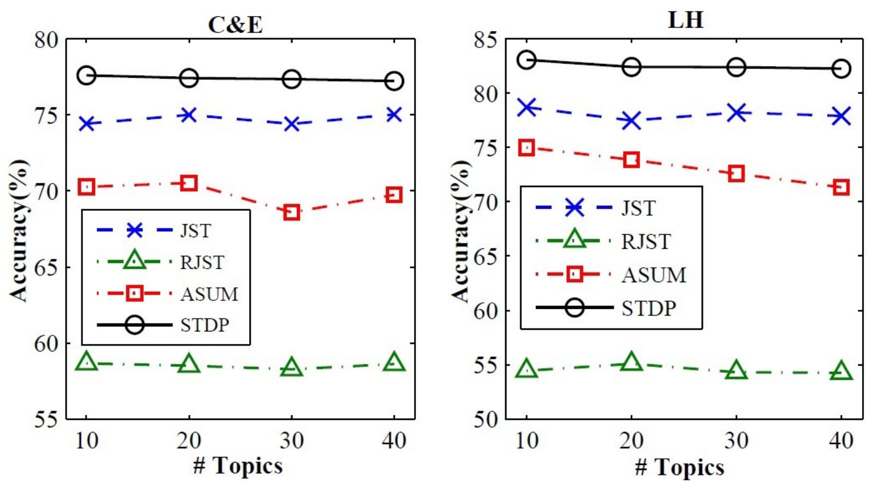


**Figure 2.20:** Sentiment prediction accuracy as the number of topics varies on data sets Computers & Electronics (C&E) and Local Restaurants and Hotels (LH).

## 2.5.3 Modelling Topic Weights

In [45], Wang et al. define the Latent Aspect Rating Analysis (LARA) problem: given a set of reviews and their overall ratings, detect topics in the review, the sentiments expressed about those topics and the relative emphasis or weight placed on each topic. Latent Rating Regression (LRR) was their initial solution and did not explicitly model topics. Instead, Wang et al. used a set of seed words and an EM-like bootstrap algorithm to classify sentences into topics and then applied the LRR model for sentiment analysis and detecting topic weights. Latent Aspect Rating Analysis Model (LARAM) [46], developed by the same authors, was an extension to LRR that incorporated topics into the model itself.

The topic classification algorithm used in LRR is shown in Figure 2.21. The output of the algorithm is a word frequency matrix $W_d$ for each document containing the normalized frequencies of words in each aspect.

**Input:** A collection of reviews $\{d_1, d_2, \ldots, d_{|D|}\}$, set of aspect keywords $\{T_1, T_2, \ldots, T_k\}$, vocabulary V, selection threshold p and iteration step limit I.
**Output:** Reviews split into sentences with aspect assignments.
**Step 0:** Split all reviews into sentences, $X = \{x_1, x_2, \ldots, x_M\}$;
**Step 1:** Match the aspect keywords in each sentence of X and record the matching hits for each aspect i in Count(i);
**Step 2:** Assign the sentence an aspect label by $a_i$ = argmaxi Count(i). If there is a tie, assign the sentence with multiple aspects;
**Step 3:** Calculate $\chi^2$ measure of each word (in V);
**Step 4:** Rank the words under each aspect with respect to their $\chi^2$ value and join the top p words for each aspect into their corresponding aspect keyword list $T_i$;
**Step 5:** If the aspect keyword list is unchanged or iteration exceeds I, go to Step 6, else go to Step 1;
**Step 6:** Output the annotated sentences with aspect assignments;

**Figure 2.21:** Bootstrapping algorithm for LRR.

Once the algorithm has been applied to the data set the aspect sentiments are assumed to be generated as a linear combination of $W_d$ and s:

$$\eta_i = \sum_{j=1}^{n} s_{i,j} W_{d,i,j} \tag{2.18}$$

where $s_i$ is the word sentiment polarity on aspect $z_i$. We then draw a prior for the aspect weights from a Gaussian distribution:

$$\kappa_i \sim N(\mu, \Sigma) \tag{2.19}$$

where μ and Σ are mean and variance parameters respectively. The overall rating is then represented as a sample drawn from a Gaussian distribution with mean $\kappa_d^T \eta_d$ and variance ν:

$$r_d \sim N\left(\sum_{i=1}^{T} \kappa_{d,i} \sum_{j=1}^{N} s_{i,j} W_{d,i,j}, \nu^2\right) \tag{2.20}$$

Combining everything together, the probability of the overall rating for a given document is:

$$P(r|d)=\int p(\kappa_d|\mu,\Sigma)\,p(r|\sum_{i=1}^{T}\kappa_{d,i}\sum_{i=1}^{N}s_{d,i,j}W_{d,i,j},\nu^2)\,d\kappa_d \tag{2.21}$$

Wang et al. used a maximum a posteriori (MAP) estimation method to estimate the aspect weights given parameters μ, Σ, $\nu^2$ and s and a maximum likelihood estimator to estimate those parameters.

In LARAM, Wang et al. replaced the bootstrap algorithm with an LDA-like component; each topic was modelled by a multinomial distribution over the vocabulary and the topic distribution in each review is drawn from a Dirichlet distribution. The probability of the overall rating and the observed text content W (not to be confused with $W_d$ from LRR) is then:

$$\begin{aligned} p(r,W|\phi,\alpha,\delta,\mu,\Sigma,\nu^2)=\int\int\prod_{n=1}^{|d|}\sum_{z_n=1}^{T}p(w_n|z_n)\,p(z_n|\theta)\,p(\theta|\alpha)\,d\theta \\ \times\ p(r|\sum_{i=1}^{T}\kappa_i\sum_{n=1}^{|d|}s_{i,j}\Delta[w_n=v_j,z_n=i],\nu^2)\,p(\kappa|\mu,\Sigma)\,d\kappa \end{aligned} \tag{2.22}$$

Where Δ – an indicator function – replaces the word frequency matrix $W_d$. To infer the posterior distribution of the topic weights and topic sentiments Wang et al. used variational approximation. The two models are depicted graphically in Figure 2.22 along with LDA for comparison.

These are the first models to our knowledge to introduce the notion of topic weights. However, these models are somewhat incomplete as they require the overall rating to be given. In our work, we do not make this assumption but rather try to infer the overall rating from the data.

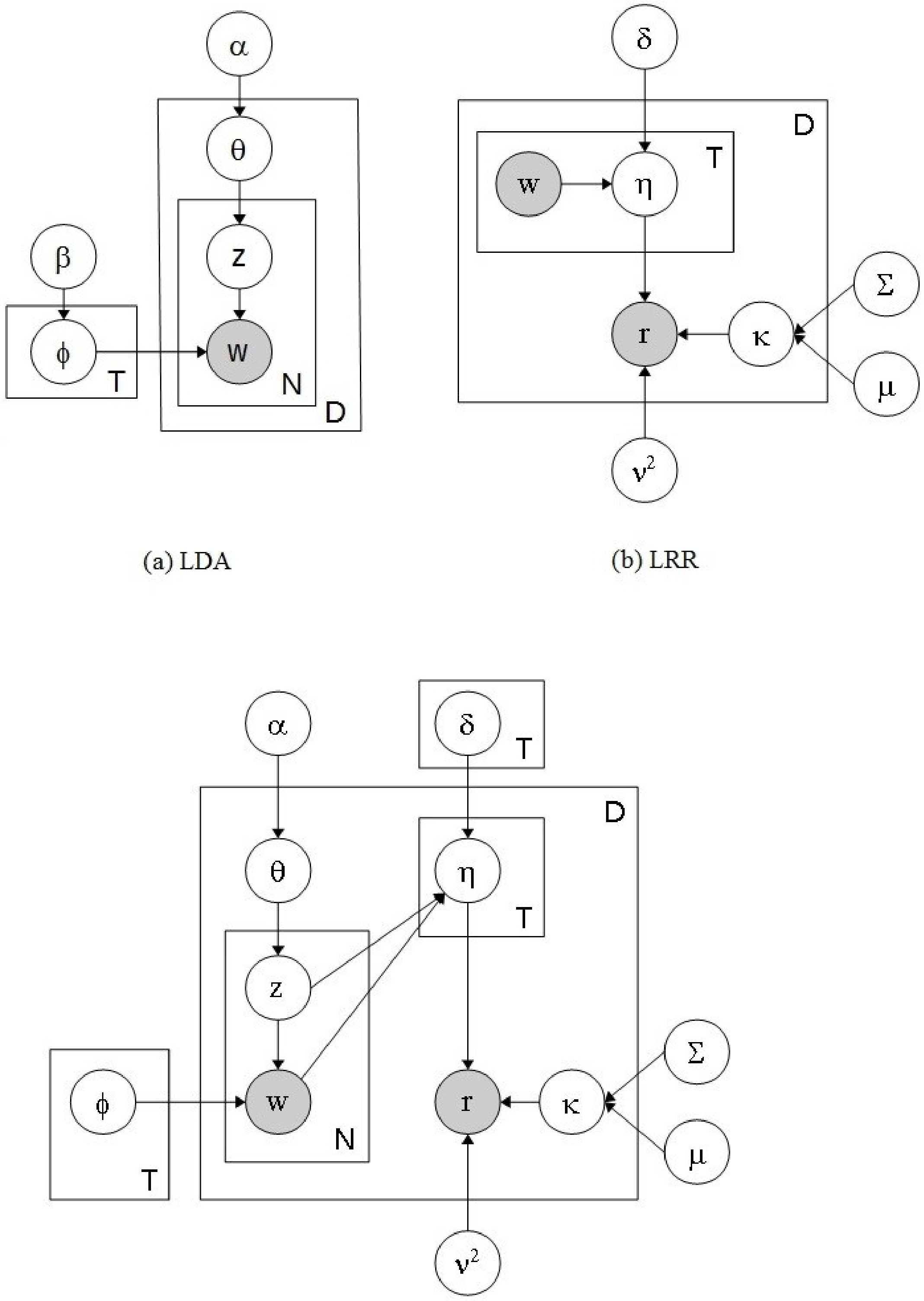


**Figure 2.22:** LDA, LRR and LARAM models

# Chapter 3

# Proposed Solution

In this chapter we present our SentPOSLDA model and explain the details of its inference and optimizations. We also describe our parallel implementation of the Gibbs sampler for faster processing along with the reasoning behind it.

## 3.1 SentPOSLDA Model

POSLDA [9], as introduced in Chapter 2, can be seen as the starting point of our model. We model topics and syntax in the same way; LDA [7] is the basis for the topic modelling component and the Bayesian HMM is the basis for the syntax modelling component.

We add a component to model the probability of a word being a sentiment word or a non-sentiment word based on its POS class, borrowing the ideas from Li, et al's STDP [24]. However, instead of using a POS tagger, we take advantage of the syntax modelling component of our model. Furthermore, we restrict this component to operate on semantic classes only, since the syntax classes do not normally capture sentiment-expressing words.

To add a sentiment analysis component we draw inspiration from JST [25]. However, whereas JST has a topic distribution for each sentiment label, our model has a sentiment distribution for each topic. Intuitively, this is a more appropriate approach for what we are trying to achieve. A topic distribution for each sentiment label answers the question “which topics does the author like/dislike more than others?” whereas our approach answers the question “how much does the author like each topic?”. We also do not explicitly model an overall sentiment that influences aspect sentiments but rather derive the overall sentiment from the individual sentiments of the underlying aspects.

To get the overall sentiment for a document we make the simplifying assumption that the more a person talks about a topic the more important that topic is to that person. Under this assumption, the topic proportion θ is equivalent to the topic weights in LRR/LARAM [45], [46] and we can thus derive the overall sentiment in a similar way.

Intuitively, SentPOSLDA simulates the following process a user would follow to write a review:

1. The user decides to write a review describing a set of topics. She also decides how much she likes/dislikes each topic.
2. She first selects a POS group for the next word she will use, following the rules of syntax.
3. Depending on the POS group selected, she makes a choice over expressing something topic-related or using a functional word to form a bigger phrase.
    (a) If she decides to use a functional word she picks the word from the selected syntax class and goes back to step 2.
    (b) Otherwise, based on the POS group, she chooses to either identify a topic or express a sentiment about a topic.
        i. If she chooses to identify a topic, she picks the topic to identify. She then picks a word to identify that topic and goes back to step 2.

ii. Otherwise, she picks a sentiment associated with a topic she mentioned in the current sentence. She then picks a word to express that sentiment and goes back to step 2.

Notice that in step 3.b.ii we further restrict the expression of a sentiment to a topic that was mentioned in the current sentence. Such dependence between a sentiment word and its target is something that is missing in all the models we described so far. ASUM [22] is the only one that can somewhat mimic that as it operates at the level of a sentence.

The formal generative process for SentPOSLDA is as follows:

1. For each row $\pi_r$ in $\pi$ for the POS classes:

    (a) Draw $\pi_r \sim$ Dirichlet($\gamma$)

2. For each semantic POS class c:

    (a) Draw $\rho_c \sim$ Beta($\varepsilon$)

3. For each word distribution $\phi_t \in \phi$:

    (a) Draw $\phi_t \sim$ Dirichlet($\beta$)

4. For each document d:

    (a) Draw topic distribution $\theta_d \sim$ Dirichlet($\alpha$)

    (b) For each topic z:

        i. Draw sentiment distribution $\eta_{d,z} \sim$ Dirichlet($\delta$)

    (c) For each word token $w_i$ in document d:

        i. Draw POS class $c_i \sim \pi_{c_{i-1}}$

```
ii. If $c_i \leq C_{SEM}$:
    A. Draw word $w_i \sim \phi_{ci}^{(SYN)}$
iii. Else:
    A. Draw topic $z_i \sim \theta_d$
    B. Draw sentiment/non-sentiment indicator $f_i \sim \rho_{ci}$
    C. If $f_i$ = true
        I. Draw sentiment $s_i \sim \eta_{d,zi}$
        II. Draw word $w_i \sim \phi_{ci,zi,si}^{(SEM)}$
    D. Else:
        I. Draw word $w_i \sim \phi_{ci,zi}^{(SEM)}$
```

As described, our model keeps track of topic and sentiment distributions on a per-document basis. However, it can easily be converted to do that on a per-sentence basis. To do that, we treat each sentence the same way we have been treating documents so far. Then, after we perform inference and wish to derive document-level properties, we simply aggregate appropriate distributions for all sentences within a document. The details of how to do that will be presented in the next section.

The reason to switch between document-level or sentence-level models is that one may be more appropriate over the other based on the nature of the corpus and the level at which we want to detect sentiment. We posit that a sentence-level approach is more appropriate in cases where each document contains several fine-grain aspects and we want to detect sentiment at the aspect level. Conversely, we claim that a document-level approach would work better when each document contains one overall topic and we want to detect overall sentiment for each document.

A graphical representation of the model is shown in Figure 3.1.

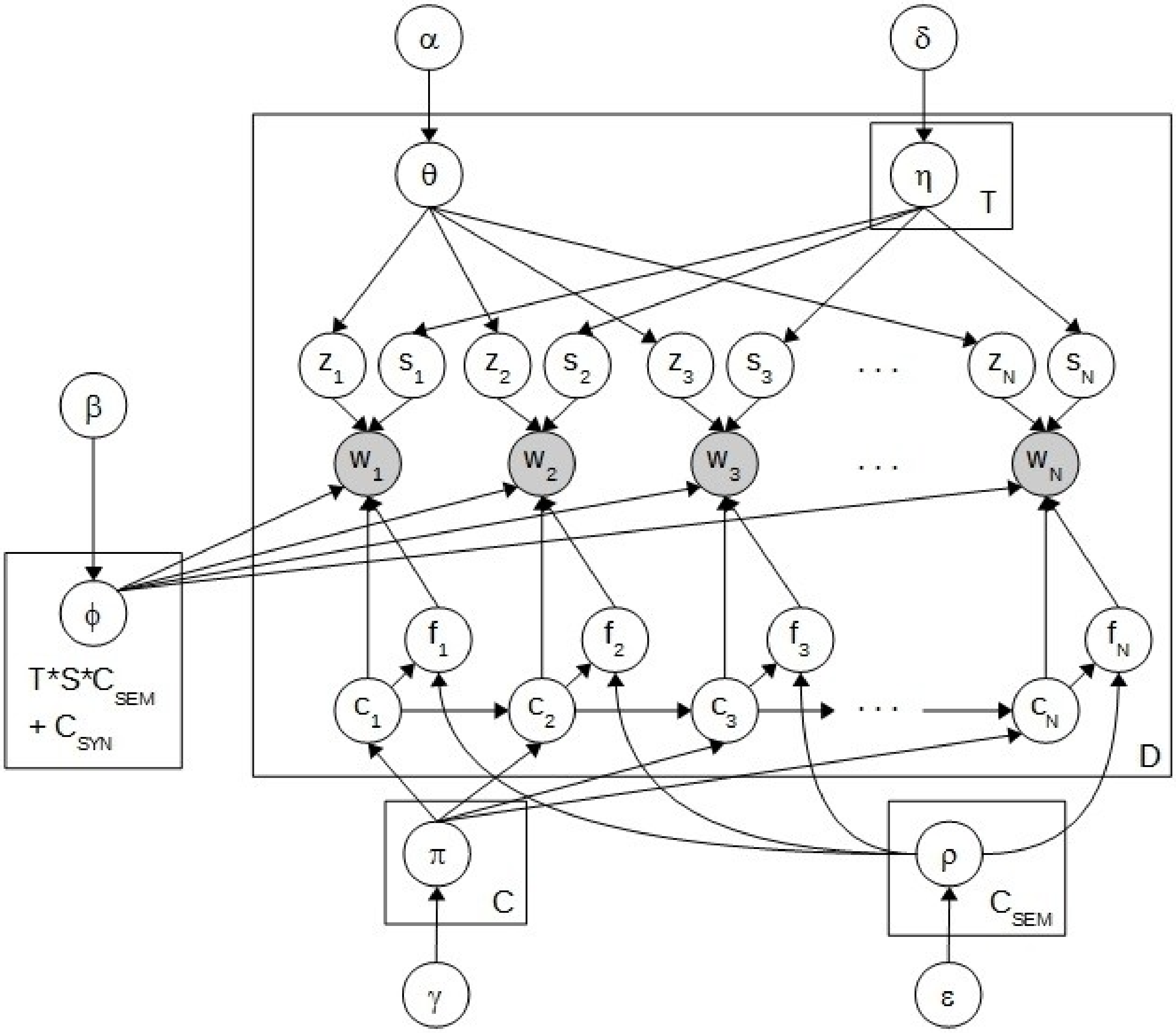


**Figure 3.1:** Graphical representation of the SentPOSLDA model

## 3.2 Inference

As with the models described in Chapter 2, the task at hand is posterior inference and our method of choice is Gibbs sampling. Our sampler needs to handle three cases:

1. The observed word is in a syntactic class.
2. The observed word is in a semantic class but does not express a sentiment.
3. The observed word is in a semantic class and expresses a sentiment.

Following Darling in [9], we begin by deriving the Gibbs sampling equations for each of the components separately – the class $c_i$, topic $z_i$, sentiment $s_i$ and sentiment/non-sentiment indicator $f_i$ – and then we combine the results into a single framework. The derivation for the POS classes of the Gibbs sampling process has already been described in section 2.3. The resulting equation (2.14) covers case 1 above. Similarly, the derivation for the topics is the same as for LDA and POSLDA and can be found in [8], [9] and [43]. Equation (2.10) gives us the desired result for the topics and equations (2.16) and (2.17) together cover the combination of POS classes and topics.

Recall that our model has a sentiment distribution for each topic. To derive the sentiment portion of the sampler it is helpful to start with the simplest case where there is only one topic. In this case, the sentiment modelling component in SentPOSLDA is identical to the topic modelling component. The desired equation for Gibbs sampling then becomes identical to (2.10) with δ in place of α and s in place of z:

$$p(s_i|s_{-i}, w_i) \propto \frac{n_{w_i}^{(s_i)}+\beta}{n^{(s_i)}+W\beta} \cdot \frac{n_{s_i}^{(d)}+\delta_{s_i}}{n^{(d)}+\delta} \tag{3.1}$$

When extended to multiple topics, the counts in the above equation are restricted to the words assigned to a given topic:

$$p_{z_i}(s_i|s_{-i},w_i) \propto \frac{n_{w_i}^{(z_i,s_i)}+\beta}{n^{(z_i,s_i)}+W\beta}\cdot\frac{n_{s_i}^{(d,z_i)}+\delta_{s_i}}{n^{(d,z_i)}+\delta} \tag{3.2}$$

Combining (3.2) with the semantic case of equation (2.17) gives us the probability distribution for class $c_i$, topic $z_i$ and sentiment $s_i$:

$$p(c_i,z_i,s_i|c_{-i},z_{-i},s_{-i},w) \propto p_{c_i}\cdot\frac{n_{w_i}^{(c_i,z_i,s_i)}+\beta}{n^{(c_i,z_i,s_i)}+W\beta}\cdot\frac{n_{z_i}^{(d)}+\alpha_{z_i}}{n^{(d)}+\alpha}\cdot\frac{n_{s_i}^{(d,z_i)}+\delta_{s_i}}{n^{(d,z_i)}+\delta} \tag{3.3}$$

The remaining component is the sentiment/non-sentiment indicator that will be used to distinguish between cases 2 and 3. Our model has a separate distribution for each POS class; once again, it is helpful to start with the case of only one class. More formally, we need the following posterior distribution:

$$p(f_i|f_{-i},\epsilon) \tag{3.4}$$

The definition of conditional probability tells us that:

$$p(f_i|f_{-i},\epsilon)=\frac{p(f_i,f_{-i}|\epsilon)}{p(f_{-i},\epsilon)}\propto p(f_i,f_{-i}|\epsilon)=p(f_i|\epsilon) \tag{3.5}$$

We then have:

$$p(f|\epsilon)=\int p(f,\rho|\epsilon)d\rho \tag{3.6}$$

We can expand the above equation based on the definition of our model:

$$p(f|\epsilon)=\int p(\rho|\epsilon)p(f|\rho)d\rho \tag{3.7}$$

We can now expand and simplify starting with the case of $f_i$ = true:

$$\begin{aligned}
p(f_i{=}true|\epsilon) &= \int p(\rho|\epsilon)\rho\, d\rho \\
&= \int \frac{\Gamma(n+\epsilon-1)}{\Gamma(n_{f_i=true}+\epsilon)\Gamma(n_{f_i=false}+\epsilon)} \rho^{n_{f_i=true}+\epsilon_{true}-1}(1-\rho)^{n_{f_i=false}+\epsilon_{false}-1}\rho\, d\rho \\
&= \frac{\Gamma(n+\epsilon-1)}{\Gamma(n_{f_i=true}+\epsilon)\Gamma(n_{f_i=false}+\epsilon)} \int \rho^{n_{f_i=true}+\epsilon_{true}}(1-\rho)^{n_{f_i=false}+\epsilon_{false}-1} d\rho \\
&= \frac{\Gamma(n+\epsilon-1)}{\Gamma(n_{f_i=true}+\epsilon_{true})\Gamma(n_{f_i=false}+\epsilon_{false})} \frac{\Gamma(n_{f_i=true}+\epsilon_{true}+1)\Gamma(n_{f_i=false}+\epsilon_{false})}{\Gamma(n+\epsilon)} \\
&= \frac{\Gamma(n+\epsilon-1)}{\Gamma(n_{f_i=true}+\epsilon_{false})} \frac{\Gamma(n_{f_i=true}+\epsilon_{true}+1)}{\Gamma(n+\epsilon)}
\end{aligned} \tag{3.8}$$

where Γ(·) is the gamma function. Using the fact that Γ(x+1) = xΓ(x):

$$\begin{aligned}
p(f_i{=}true|\epsilon) &= \frac{\Gamma(n+\epsilon-1)}{\Gamma(n_{f_i=true}+\epsilon_{true})} \frac{\Gamma(n_{f_i=true}+\epsilon_{true}+1)}{\Gamma(n+\epsilon)} \\
&= \frac{\Gamma(n+\epsilon-1)}{\Gamma(n_{f_i=true}+\epsilon_{true})} \frac{\Gamma(n_{f_i=true}+\epsilon_{true}+1)}{(n+\epsilon-1)\Gamma(n+\epsilon-1)} \\
&= \frac{\Gamma(n_{f_i=true}+\epsilon_{true}+1)}{\Gamma(n_{f_i=true}+\epsilon)(n+\epsilon-1)} \\
&= \frac{(n_{f_i=true}+\epsilon_{true})\Gamma(n_{f_i=true}+\epsilon_{true})}{\Gamma(n_{f_i=true}+\epsilon_{true})(n+\epsilon)} \\
&= \frac{n_{f_i=true}+\epsilon_{true}}{n+\epsilon}
\end{aligned} \tag{3.9}$$

Extending this to multiple classes, the probability for each class is as follows:

$$p_c(f_i{=}true|\epsilon) \propto \frac{n^{(c)}_{f_i=true}+\epsilon_{true}}{n^{(c)}+\epsilon} \tag{3.10}$$

Similarly,

$$p_c(f_i = false|\epsilon) \propto \frac{n^{(c)} f_i = false + \epsilon_{false}}{n^{(c)} + \epsilon} \tag{3.11}$$

Finally, putting (2.16), (2.17), (3.3), (3.10) and (3.11) together we get a complete set of equations for Gibbs sampling, corresponding to the three cases identified at the beginning of this subsection:

1. $c_i > C_{SEM}$:

$$p(c_i|c_{-i}, w) \propto p_{c_i} \cdot \frac{n_{w_i}^{(c_i)} + \beta}{n^{(c_i)} + W\beta} \tag{3.12}$$

2. $c_i \le C_{SEM}$ and $f_i$ = false:

$$p(c_i, z_i, f_i|c_{-i}, z_{-i}, f_{-i}, w) \propto p_{c_i} \cdot \frac{n_{z_i}^{(d)} + \alpha_{z_i}}{n^{(d)} + \alpha} \cdot \frac{n_{w_i}^{(c_i, z_i)} + \beta}{n^{(c_i, z_i)} + W\beta} \cdot \frac{n_{f_i}^{(c_i)} + \epsilon_{f_i}}{n^{(c_i)} + \epsilon} \tag{3.13}$$

3. $c_i \le C_{SEM}$ and $f_i$ = true:

$$p(c_i, z_i, s_i, f_i|c_{-i}, z_{-i}, s_{-i}, f_{-i}, w) \propto p_{c_i} \cdot \frac{n_{z_i}^{(d)} + \alpha_{z_i}}{n^{(d)} + \alpha} \cdot \frac{n_{w_i}^{(c_i, z_i, s_i)} + \beta}{n^{(c_i, z_i, s_i)} + W\beta} \cdot \frac{n_{s_i}^{(d, z_i)} + \delta_{s_i}}{n^{(d, z_i)} + \delta} \cdot \frac{n^{(c_i)}, f_i + \epsilon_{f_i}}{n^{(c_i)} + \epsilon} \tag{3.14}$$

where $p_{c_i}$ is defined in (2.16) and is repeated here for completeness:

$$p_{c_i} = \frac{n_{(c_{i-2}, c_{i-1}, c_i)} + \gamma_{c_i}}{n_{(c_{i-2}, c_{i-1})} + \gamma} \cdot \frac{n_{(c_{i-1}, c_i, c_{i+1})} + \gamma_{c_i}}{n_{(c_{i-1}, c_i)} + \gamma} \cdot \frac{n_{(c_i, c_{i+1}, c_{i+2})} + \gamma_{c_i}}{n_{(c_i, c_{i+1})} + \gamma} \tag{3.15}$$

After sufficient burn-in time for Gibbs sampling, we can estimate several probabilities. The approximate probability of a topic in a document is:

$$\theta_{d, z_i} = \frac{n_{z_i}^{(d)} + \alpha_{z_i}}{n^{(d)} + \alpha} \tag{3.16}$$

The approximate probability of a sentiment expressed for a topic in a document is:

$$\eta_{z_i,s_i,d} = \frac{n_{s_i}^{(d,z_i)} + \delta_{s_i}}{n^{(d,z_i)} + \delta} \tag{3.17}$$

For the approximate probability of a POS class expressing sentiment we have:

$$\rho_{c_i} = \frac{n_{f_{true}}^{(c_i)} + \epsilon_{true}}{n^{(c_i)} + \epsilon} \tag{3.18}$$

The approximate probability of a word with a POS class specifying a topic is:

$$\phi_{c_i,z_i,w_i} = \frac{n_{w_i}^{(c_i,z_i)} + \beta}{n^{(c_i,z_i)} + W\beta} \tag{3.19}$$

Finally, the approximate probability of a word with a POS class expressing a sentiment about a topic is:

$$\phi_{c_i,z_i,s_i,w_i} = \frac{n_{w_i}^{(c_i,z_i,s_i)} + \beta}{n^{(c_i,z_i,s_i)} + W\beta} \tag{3.20}$$

To perform the analysis at the level of a sentence the counts in the above equations need to be summed over the individual sentences within each document. In particular, equation (3.17) becomes:

$$\eta_{z_i,s_i,d} = \frac{\Sigma_{r \in R_d} n_{s_i}^{(r,z_i)} + \delta_{s_i}}{\Sigma_{r \in R_d} n^{(r,z_i)} + \delta} \tag{21}$$

where $R_d$ is the set of sentences in document d.

## 3.3 Hyperparameter Optimization

Although hyperparameters can be set manually and kept fixed, it is often time consuming to find the optimal values that way. In our work we optimize them using Minka's fixed-point method [28]. This method defines an iterative process that converges to a new, optimal value for the hyperparameter based on the current variable assignments in the data set.

The iterative process for the α hyperparameter has been derived in [42] and is defined by:

$$\alpha_z' = \alpha_z \frac{\Sigma_{d=1}^{D} \Psi(n_z^{(d)} + \alpha_z) - \Psi(\alpha_z)}{\Sigma_{d=1}^{D} \Psi(n^{(d)} + \alpha) - \Psi(\alpha)} \tag{3.22}$$

where Ψ is the digamma function. We can derive the iterative process for the δ hyperparameter in a similar fashion. The likelihood of the training set given the hyperparameter δ is:

$$p(\mathrm{X}|\delta) = \prod_{d=1}^{D} \prod_{z=1}^{T} \frac{\Gamma(\delta)}{\Gamma(n^{(d,z)} + \delta)} \prod_{s=1}^{S} \frac{\Gamma(n_s^{(d,z)} + \delta_s)}{\Gamma(\delta_s)} \tag{3.23}$$

where X are the current sentiment-word assignments. Taking log-likelihood on both sides we get:

$$\log p(\mathrm{X}|\delta) = \sum_{d=1}^{D} \sum_{z=1}^{T} [\log \Gamma(\delta) - \log \Gamma(n^{(d,z)} + \delta) + \sum_{s=1}^{S} \log \Gamma(n_s^{(d,z)} + \delta_s) - \log \Gamma(\delta_s)] \tag{3.24}$$

which can be bounded from below using the following bounds:

$$\log \Gamma(r) - \log \Gamma(r+n) \geq \log \Gamma(\tilde{r}) - \log \Gamma(\tilde{r}+n) + [\Psi(\tilde{r}+n) - \Psi(\tilde{r})](\tilde{r} - r) \tag{3.25}$$

and

$$\log \Gamma(r+n) - \log \Gamma(r) \geq \log \Gamma(\tilde{r}+n) - \log \Gamma(\tilde{r}) + \tilde{r}[\Psi(\tilde{r}+n) - \Psi(\tilde{r})](\log r - \log \tilde{r}) \tag{3.26}$$

where r is a true real number, $\tilde{r}$ is an approximate real number and n is a positive integer.

Substituting (3.25) and (3.26) into (3.24) we get:

$$\begin{aligned}\log p(X|\delta') &\geq B(\delta') \\ &= \sum_{d=1}^{D}\sum_{z=1}^{T}\Big[\log\Gamma(\delta) - \log\Gamma(n^{(d,z)}+\delta) + [\Psi(n^{(d,z)}+\delta)-\Psi(\delta)](\delta-\delta') \\ &\quad + \sum_{s=1}^{S}\log\Gamma(n_s^{(d,z)}+\delta_s) - \log\Gamma(\delta_s) \\ &\quad + \delta_s[\Psi(n_s^{(d,z)}+\delta_s)-\Psi(\delta_s)](\log(\delta_s')-\log(\delta_s))\Big]\end{aligned} \tag{3.27}$$

Now, by grouping all terms not involving δ' into a constant term C, we get:

$$\begin{aligned}\log p(X|\delta') \geq \sum_{d=1}^{D}\sum_{z=1}^{T}\Big[&[\Psi(n^{(d,z)}+\delta)-\Psi(\delta)](-\delta') \\ &+\sum_{s=1}^{S}\delta_s[\Psi(n_s^{(d,z)}+\delta_s)-\Psi(\delta_s)](\log(\delta_s'))+C\Big]\end{aligned} \tag{3.28}$$

Taking the derivative with respect to δ' we get:

$$\frac{\partial B(\delta')}{\partial \delta_s'} = \sum_{d=1}^{D}\sum_{z=1}^{T}\left[\frac{\delta_s[\Psi(n_s^{(d,z)}+\delta_s)-\Psi(\delta_s)]}{\delta_s'} - [\Psi(n^{(d,z)}+\delta)-\Psi(\delta)]\right] \tag{3.29}$$

Finally, by setting (3.29) to zero and solving for δ' we arrive at the iterative process for δ:

$$\delta_s' = \delta_s \frac{\Sigma_{d=1}^{D}\Sigma_{z=1}^{T}\Psi(n_s^{(d,z)}+\delta_s)-\Psi(\delta_s)}{\Sigma_{d=1}^{D}\Sigma_{z=1}^{T}\Psi(n^{(d,z)}+\delta)-\Psi(\delta)} \tag{3.30}$$

Following a similar methodology, we can derive the iterative process for the remaining hyperparameters:

$$\beta' \;=\; \beta\;\frac{\begin{aligned}&\left[\Sigma_{c=1}^{C_{SEM}}\Sigma_{z=1}^{T}\Sigma_{s=1}^{S}\Psi(n_w^{c,z,s}+\beta)-\Psi(\beta)\right] \\ &+\left[\Sigma_{c=1}^{C_{SEM}}\Sigma_{z=1}^{T}\Psi(n_w^{c,z}+\beta)-\Psi(\beta)\right] \\ &+\left[\Sigma_{c=S_{SEM}+1}^{C_{SYN}}\Psi(n_w^{c}+\beta)-\Psi(\beta)\right]\end{aligned}}{\begin{aligned}&\left[\Sigma_{c=1}^{C_{SEM}}\Sigma_{z=1}^{T}\Sigma_{s=1}^{S}\Psi(n^{c,z,s}+W\beta)-\Psi(W\beta)\right] \\ &+\left[\Sigma_{c=1}^{C_{SEM}}\Sigma_{z=1}^{T}\Psi(n^{c,z}+W\beta)-\Psi(W\beta)\right] \\ &+\left[\Sigma_{c=C_{SEM}+1}^{C_{SYN}}\Psi(n^{c}+W\beta)-\Psi(W\beta)\right]\end{aligned}} \tag{3.31}$$

$$\gamma_c{}' = \gamma_c \frac{\sum_{i=1}^{C} \sum_{j=1}^{C} \sum_{k=1}^{C} \sum_{m=1}^{C} \Psi(n_{i,j,k,m} + \gamma_c) - \Psi(\gamma_c)}{\sum_{i=1}^{C} \sum_{j=1}^{C} \sum_{k=1}^{C} \Psi(n_{i,j,k} + \gamma_c) - \Psi(\gamma_c)} \tag{3.32}$$

$$\epsilon_f{}' = \epsilon_f \frac{\sum_{c=1}^{C_{SEM}} \Psi(n_f^{(c)} + \epsilon_f) - \Psi(\epsilon_f)}{\sum_{c=1}^{C_{SEM}} \Psi(n^{(c)} + \epsilon) - \Psi(\epsilon)} \tag{3.33}$$

## 3.4 Seeding

Our model is designed to classify sentiment words into groups based on co-occurrence statistics so that words with similar sentiment can be identified. However, the model itself does not have the concepts of "good" or "bad" built into it. In order to coerce SentPOSLDA to group all negative sentiment words into one specific category and all positive sentiment words into another, we add seed words for both positive and negative sentiments. As we explained in Chapter 1, choosing a list of words that express a certain sentiment is not a trivial task as sentiment is context and domain dependent. We therefore have to choose our list carefully to ensure that each word is as independent of context and domain as possible. Readers are referred to Appendix A for lists of seed words for positive and negative sentiment.

Since, for some of our experiments, we are interested in testing the model against a data set that gives us the author's sentiment ratings for a set of aspects, it is necessary to coerce the model to classify words into specific topic categories. We do that using a separate set of topic seed words. Appendix B shows the topic seed words for the aspects of one of our data sets.

There are different ways of incorporating seed words into a probabilistic model. One approach is to adjust the beta hyperparameter whenever we encounter a seed word to give that word a higher chance of staying in the desired category. A stronger approach, the one we use for our model, is to “pin” the seed words in the desired categories for the duration of the sampling process so that their assignments never change.

It is important to note that adding seed words changes our solution from an unsupervised to a semi-supervised one. Without seed words one could still use our solution but some manual post-processing analysis would be necessary to figure out which sentiment category is positive or negative and which topics are represented in the topic categories. Furthermore, the topics discovered might be slightly different from the ones we are interested in. Measuring the quality of results would be much more difficult, if not impossible, in that case so for the purposes of our experiments we deem such a compromise an acceptable one.

## 3.5 Post-Processing

### 3.5.1 Overall Sentiment

SentPOSLDA does not explicitly model the overall sentiments for a document. As explained in section 3.1, we make the simplifying assumption that the more a person talks about a topic the more important that topic is to that person. Under that assumption the topic proportions are analogous to topic weights in LRR [45] and LARAM [46] models. We can therefore derive the overall sentiment for a document as follows:

$$Overall_{d,s} = \frac{\sum_{z=1}^{T} \theta_z \eta_{z,s,d}}{\sum_{i=1}^{S} \sum_{z=1}^{T} \theta_z \eta_{z,i,d}} \qquad (3.34)$$

Our assumption is, of course, not always true. It is entirely possible to encounter a review such as the following:

> "The service at this restaurant was exceptional. The waiter was very attentive and friendly. He worked really hard to make our experience enjoyable. Unfortunately, the food was horrible so I have to give it a one star."

The reviewer talked about the service for about 3/4 of the review but at the very end we find out that the horrible food had outweighed the excellent service so much that it dominated the overall sentiment. Nevertheless, we hypothesize that the degree of inaccuracy introduced by our assumption is negligible.

### 3.5.2 Undetected Sentiments

Consider the following hotel review from the TripAdvisor web site [22]:

| | | | |
|---|---|---|---|
| Content | The location of this hotel is great! We have stayed there with and without a car. If we don't want to do everything together it is easy for us to go our separate ways and connect back up. The staff was also very friendly and the hotel very clean! | | |
| Overall | 5 | Cleanliness | 5 |
| Value | 5 | Service | 5 |
| Rooms | 5 | Front Desk | 5 |
| Location | 5 | Business Service | 5 |

The reviewer mentioned the location, the staff and cleanliness but did not mention the other four aspects. In our model, we would not detect any sentiments for the aspects that were not mentioned and the counts in equation (3.17) would all equal to 0 for those aspects. Consequently, the equation would be reduced to $\delta_s/\delta$, which is roughly the overall sentiment for the entire corpus. Obviously, this is a bad estimate. A more appropriate estimate is the overall sentiment of the review. We therefore calculate the overall sentiment first, using equation (3.34), ignoring any aspects that were not mentioned – and then use the overall sentiment as an estimate for those missing aspects.

## 3.6 Concurrent Sampling

SentPOSLDA is a rather complex model requiring a large amount of computations to perform the inference task. In order to keep the efficiency at a reasonable level we have developed a multi-threaded Gibbs sampling algorithm for SentPOSLDA that allows us to cut down the running time of an experiment from days to hours. Recall from Chapter 2 that the standard form of the Gibbs sampling algorithm performs inference as follows:

```
1. Randomly initialize $\{c_1, z_1, f_1, s_1\}, ..., \{c_N, z_N, f_N, s_N\}$ for all words $w_1, ..., w_N$ in a corpus.
2. For $t = 1, ..., I$:
   (a) For $i = 1, ..., N$:
       i.   Decrement all counts involving $w_i$, $c_i$, $z_i$, $f_i$, and $s_i$
       ii.  Calculate $p(\{c_i, z_i, f_i, s_i\} \mid \{c_1, z_1, f_1, s_1\}, ..., \{c_{i-1}, z_{i-1}, f_{i-1}, s_{i-1}\}, \{c_{i+1}, z_{i+1}, f_{i+1}, s_{i+1}\}, ..., \{c_N, z_N, f_N, s_N\}, w_i)$
       iii. Draw $\{c_i, z_i, f_i, s_i\} \sim p(\{c_i, z_i, f_i, s_i\} \mid \{c_1, z_1, f_1, s_1\}, ..., \{c_{i-1}, z_{i-1}, f_{i-1}, s_{i-1}\}, \{c_{i+1}, z_{i+1}, f_{i+1}, s_{i+1}\}, ..., \{c_N, z_N, f_N, s_N\}, w_i)$
       iv.  Increment counts to include $w_i$, $c_i$, $z_i$, $f_i$, and $s_i$
```

where the counts are the $n_{(\cdot)}^{(\cdot)}$ values in equations (3.12) through (3.15).

To make processing of the large amount of data made available in the past decade more manageable, significant research has been done into parallel implementations of topic modelling in general and Gibbs sampling of LDA-like models in particular (ex. [30], [27], [2], [29], [26], [37], [1], [52], [3], [35], [49]). The challenge with designing a parallel implementation of the Gibbs sampling algorithm is that the algorithm is by definition sequential. When sampling from the posterior distribution, the update of the latent state of one of the words, for example its topic, cannot be done at the same time as updates for any of the other words. However, the recurring conclusion in recent research has been that when the number of words is much larger than the number of threads, the dependence of the update of one word's latent state on the update of another word's latent state is very weak. That research has been done on specific models, usually LDA, but our hypothesis is that the same result will apply to SentPOSLDA. To that end, we break up the loop in step 2a into several loops, each executing in parallel in a separate thread and each processing a subset of the corpus. Since we are modelling syntax – which is sequential in nature within a single sentence – we split the corpus into subsets at the sentence level so that all the words within a given sentence can be processed in sequence by the same thread. More formally, our algorithm becomes:

```
1. Randomly initialize $\{c_1, z_1, f_1, s_1\}, ..., \{c_N, z_N, f_N, s_N\}$ for all words $w_1, ..., w_N$ in a corpus.
2. Split corpus into P sets of M/P sentences, $Q = \{Q_1, ..., Q_P\}$
3. For t = 1, ..., I:
    (a) Execute P threads and in each thread:
        i. For $i = Q_i^{(1, start)}, ..., Q_i^{(M/P, end)}$
            1. Decrement all counts involving $w_i$, $c_i$, $z_i$, $f_i$, and $s_i$
            2. Calculate $p(\{c_i, z_i, f_i, s_i\} | \{c_1, z_1, f_1, s_1\}, ..., \{c_{i-1}, z_{i-1}, f_{i-1}, s_{i-1}\}, \{c_{i+1}, z_{i+1}, f_{i+1}, s_{i+1}$
```

$\},\ldots, \{c_N, z_N, f_N, s_N\}, w_i)$

3. Draw $\{c_i, z_i, f_i, s_i\} \sim p(\{c_i, z_i, f_i, s_i\} \mid \{c_1, z_1, f_1, s_1\}, \ldots, \{c_{i-1}, z_{i-1}, f_{i-1}, s_{i-1}\}, \{c_{i+1}, z_{i+1}, f_{i+1}, s_{i+1}\},\ldots, \{c_N, z_N, f_N, s_N\}, w_i)$
4. Increment counts to include $w_i$, $c_i$, $z_i$, $f_i$, and $s_i$

where $Q_i^{(1,\ \text{start})}$ and $Q_i^{(M/P,\ \text{end})}$ indicate the start of the first sentence and end of last sentence in $Q_i$ respectively.

With this approach, all the threads will increment and decrement the counts concurrently. To avoid issues with one thread overwriting the result of another thread's increment or decrement operation we implement atomic updates for these counts.

When there are many increment / decrement operations happening concurrently we have to be mindful of the fact that the distribution based on these counts will be slightly different than a distribution calculated by a traditional Gibbs sampling algorithm. We posit that, with the exception of very small data sets, the differences are negligible even with the modern hardware that supports thousands of threads. Intuitively, we can see why this is likely to be true by examining the terms in equations (3.12) through (3.15).

Since each thread will process one word at a time, the counts involving a word $w_i$ are only concerning if two or more threads happen to process the same word. However, even with our modest data set where the total number of words is in the order of $10^{10}$ and the number of unique words is in the order of $10^4$, the probability of this happening is very low. When it does happen, the difference in the value of the counts, even with our GPU capable of running 8129 threads, is at most $8192 / (10^{10}/10^4) = 8192/10^6 = 0.008192 < 1\%$.

Since we always split the corpus at the sentence level, it is possible that a document will be split between two sets and that two threads will be processing the same document concurrently. In this case, the counts involving a document will be updated by both threads. However, since we have preserved the word order, the first portion of the document's sentences will be at the end of one set and the remaining portion at the beginning of another set. That means that processing of the latter portion of the sentences will finish long before the processing of the first portion even begins. This case is, therefore, highly unlikely. Alternatively, we can split the corpus at the document level which would guarantee that this case never occurs. We chose a sentence-level split to distribute the workload more evenly amongst the threads.

Looking at the counts not involving a word or a document, we observe that with the magnitude of the parameters in low double-digits and the total word count being in the order of $10^{10}$, the difference in the value of the counts with our highly-multithreaded GPU is at most $8192 / (10^{10}/10^{3}) = 8192/10^{7} = 0.0008192 < 0.1\%$.

Finally, since we are sampling from a distribution similar to a one a traditional Gibbs sampler would build and then incrementing the counts with the new values at the end of each iteration, these differences are not cumulative.

# Chapter 4

# Experiments and Results

This chapter presents the results and analysis of our experiments performed with SentPOSLDA. Our primary focus is on the effectiveness of our model, and our secondary focus is on the efficiency and efficacy of our parallel Gibbs sampling algorithm.

## 4.1 Data Sets and Evaluation Methods

### 4.1.1 Data Sets

Although SentPOSLDA is a general model that captures syntax, topic and sentiment information, the goal of our thesis is to apply it to aspect-level sentiment analysis. Accordingly, the data sets for our experiments must contain opinionated text. We choose three such sources of data, each having unique characteristics to test our model against different scenarios.

The first data source is a set of hotel reviews from the TripAdvisor web site. The web site users rate hotels on up to seven aspects: value, room, location, cleanliness, check in/front desk, service and business service. For most of our experiments with TripAdvisor data we use the same sample of reviews as Wang et al. did in their experiments with LRR/LARAM [45], [46]. Comparing the performance of our model against LRR/LARAM would be an excellent way to see where SentPOSLDA ranks amongst other models. However, even though LRR/LARAM perform aspect-level sentiment analysis, the objective and the inputs and outputs of their algorithms are not the same as in our work. We therefore turn to the work of Zhou [51] who used POSLDA to separate content words from syntax words before performing sentiment analysis using a maximum entropy algorithm. The data set used by Zhou is also from TripAdvisor but contains ratings on five aspects: value, room, location, cleanliness and service. We will denote Wang et al.'s data set as TA1 and Zhou's data set as TA2. Table 4.1 shows some general statistics of the two data sets. Although their averages at word, sentence and document levels are similar, TA1 has about 20 times more reviews and a much richer vocabulary than TA2.

| | **TA1** | | **TA2** | |
|---|---|---|---|---|
| Number of reviews | 192,997 | | 15,242 | |
| Number of sentences | 2,055,033 | | 195,424 | |
| Number of words | 37,103,804 | | 3,262,753 | |
| Number of unique word tokens | 146,709 | | 28,102 | |
| | Average | Std. Dev. | Average | Std. Dev. |
| Word length (characters) | 8.07 | 2.84 | 7.32 | 2.49 |
| Sentence length (words) | 18.06 | 15.37 | 16.70 | 12.05 |
| Document length (words) | 192.25 | 212.06 | 214.06 | 178.05 |

**Table 4.1:** Properties of the two TripAdvisor data sets.

TripAdvisor allows the user to provide a rating on a scale from 1 to 5 with 1 being the worst rating and 5 being the best. To compare such a rating to the outcome of SentPOSLDA we collapse the ratings so that a rating of 1 or 2 is treated as negative and a rating of 4 or 5 as positive. We ignore all ratings of 3 since they do not indicate a positive or negative sentiment. Table 4.2 shows the distribution of ratings for the TA1 and TA2 data sets. Both data sets are heavily biased towards the positive although TA2 is less biased than TA1.

| | TA1 | | | TA2 | | |
|---|---|---|---|---|---|---|
| | # of ratings | Negative (%) | Positive (%) | # of ratings | Negative (%) | Positive(%) |
| Overall | 192,997 | 15.0 | 74.6 | 15,242 | 19.8 | 62.1 |
| Value | 149,513 | 13.7 | 71.9 | 13,308 | 20.7 | 59.0 |
| Rooms | 151,560 | 12.9 | 72.5 | 12,193 | 19.9 | 58.8 |
| Location | 115,395 | 5.7 | 83.7 | 10,880 | 5.3 | 84.2 |
| Cleanliness | 151,522 | 8.4 | 80.3 | 13,377 | 13.3 | 70.6 |
| Service | 148,490 | 11.8 | 74.9 | 13,292 | 18.4 | 63.7 |
| Front Desk | 115,493 | 10.7 | 75.0 | | | |
| Business Service | 76,465 | 12.9 | 62.9 | | | |

**Table 4.2:** Distribution of ratings in the two TripAdvisor data sets.

TripAdvisor data contains reviews on one overall topic (the hotels) and several aspects of that topic. We would like to see how SentPOSLDA does with a more complex data set that has several topics, each having several aspects. To that end we use the set of reviews from the Amazon web site that Jo & Oh used for their experiments with ASUM [22]. Each review in this data set is on one of 7 types of products: air conditioner, canister vacuum, coffee machine, digital SLR, laptop, MP3 player, and space heater. Each type of products has a different set of aspects that are not provided to us. The Amazon reviews are not structured as nicely as the TripAdvisor ones and only provide the overall ratings for each review. As a result, our performance report is limited to qualitative results and quantitative results using only the overall ratings. Table 4.3 shows some properties of the Amazon data set.

| | | |
|---|---|---|
| Number of reviews | | 24,259 |
| Number of sentences | | 268,079 |
| Number of words | | 4,428,861 |
| Number of unique word tokens | | 36,061 |
| | Average | Std. Dev. |
| Word length (characters) | 7.39 | 2.57 |
| Sentence length (words) | 16.52 | 11.42 |
| Document length (words) | 182.57 | 200.32 |

**Table 4.3:** Properties of the Amazon data set.

| | |
|---|---|
| Total ratings | 24,259 |
| Negative (%) | 20.3 |
| Positive (%) | 71.8 |

**Table 4.4:** Distributon of ratings in the Amazon data set.

Similarly to TripAdvisor, Amazon allows users to rate products on a scale of 1 to 5. We collapse these ratings the same way as we did for TripAdvisor and ignore ratings of 3. Table 4.4 shows the ratings distribution.

Lastly, we run our model against a sample of Twitter posts to see how SentPOSLDA fares with noisy data consisting of very short documents. We use the data provided at the SemEval conference [13] that contains Twitter posts manually labelled as positive, negative, neutral or objective. Table 4.5 shows the properties of this data set.

Similarly to the TripAdvisor and Amazon data sets, we ignore all neutral or objective ratings. Table 4.6 shows the rating distribution. Unlike the other data sets, this one is heavily biased toward neutral or objective.

| | | |
|---|---|---|
| Number of reviews | 5,747 | |
| Number of sentences | 10,327 | |
| Number of words | 101,813 | |
| Number of unique word tokens | 13,418 | |
| | Average | Std. Dev. |
| Word length (characters) | 6.87 | 3.08 |
| Sentence length (words) | 9.86 | 7.18 |
| Document length (words) | 17.7 | 7.4 |

**Table 4.5:** Properties of the Twitter data set.

| | |
|---|---|
| Total ratings | 5,747 |
| Negative (%) | 14.4 |
| Positive (%) | 34.3 |

**Table 4.6:** Distribution of sentiment in the Twitter data set.

### 4.1.2 Evaluation Procedure

Before diving into the performance evaluation of our model we first determine the number of Gibbs sampling iterations required for convergence. An intuitive way to do this is to track the log-likelihood and stop sampling once it stabilizes. Figure 4.1, for example, shows that the log-likelihood for the TA1 data set with 8 topics and 2 sentiment levels stabilizes around 3000 iterations. This is roughly the number of iterations used for all the experiments in this thesis.

Although Darling [9] performed experiments with POSLDA with varying number of semantic and syntactic classes, the numbers of choice for the experiments where these numbers were kept constant were 7 for semantic and 10 for syntactic. Since the focus of our work is on sentiment analysis and not on syntax modelling we used these parameters for all the experiments in this thesis.

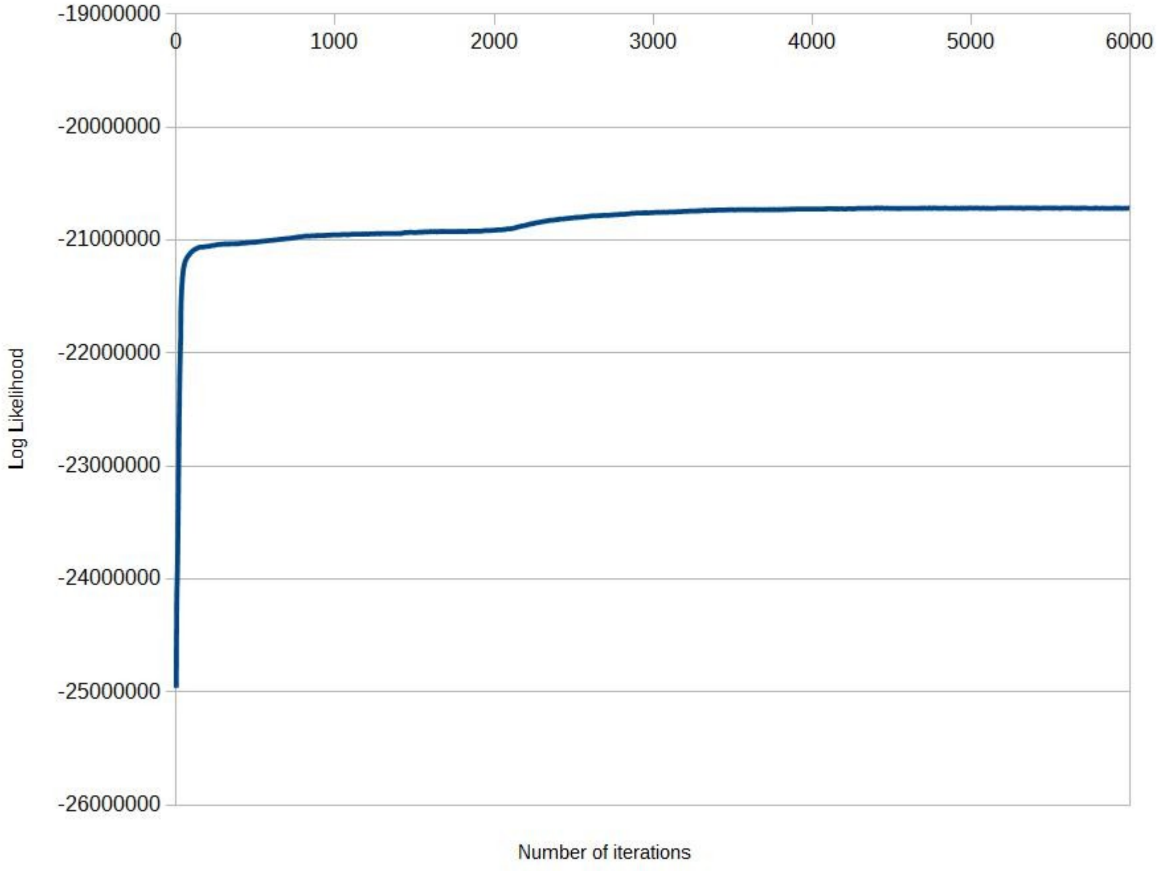


**Figure 4.1:** Log likelihood vs. number of iterations.

With the parameters in place, we run each of our experiments and use equation (3.17) to determine the aspect ratings and (3.34) for the overall rating. We then use the metrics described in the next sub-section to evaluate the performance.

For all the experiments that evaluate SentPOSLDA we use all available processing cores on our AMD HD7970 GPU to run as many threads as possible with our multi-threaded Gibbs sampler. In the final section we show that the performance of SentPOSLDA remains constant as the number of threads is decreased to prove that our results are not affected by the multi-threaded implementation.

### 4.1.3 Evaluation Metrics

We use several evaluation metrics in order to best illustrate the performance of our model. The most straightforward evaluation metric is the classification accuracy defined as the number of correct outputs divided by the total number of outputs:

$$Accuracy = \frac{\sum_{i=1}^{T} I\left(l_i = o_i\right)}{T} \tag{4.1}$$

Where T is the total number of outputs, $l_i$ are the correct, manually labelled sentiments, $o_i$ are the sentiment labels output by our model and I is an indicator function that equals to 1 when the expression inside it is true.

Classification accuracy is a good indicator in most cases. However, when the data is heavily biased towards the positive as it is with our data sets the accuracy can be misleading. To get a realistic indication of performance under these circumstances we use the Precision and Recall:

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

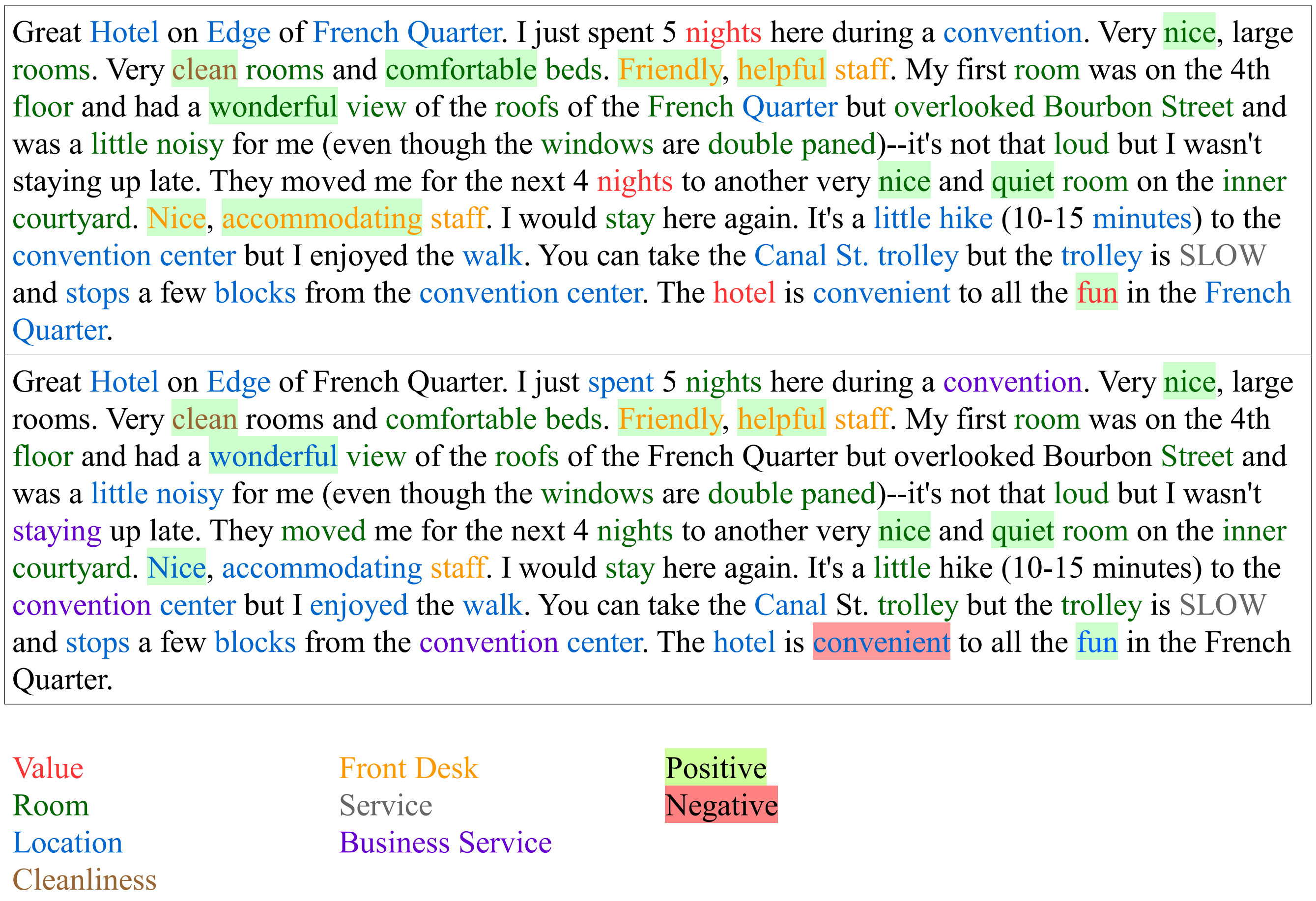
where TP is a true positive, FP is a false positive, and FN is a false negative. Zhou [51] used these same metrics in his experiments so they will be very useful for comparisons with his work.

## 4.2 SentPOSLDA Performance

### 4.2.1 Aspect-Level Sentiment Detection

The TA1 data set contains reviews on 7 aspects. However, as mentioned in Chapter 2 people often talk about things that are not quite on topic so we ran our experiments on TA1 with the number of aspects set to 8. The extra aspect is there to capture these non-rateable aspects. To force the detected aspects into specific ones that are of interest to us we used the seed lists presented in Appendix B.

As we mentioned in Chapter 3, our model can be adapted to work at either the sentence level or document level. LDA, the basis of our topic-modelling component, was designed to work with a single level of granularity. Indeed, the experiments Darling performed with POSLDA [9] – the basis of SentPOSLDA - were all with data sets that had a single topic per document and the topics POSLDA discovered were all at the document level. However, our TripAdvisor data sets contain a single topic (hotels) for all documents and each document discusses several aspects. As Figure 4.2 illustrates, the topics assigned to words when running SentPOSLDA at the document level on this data set are more noisy, with each sentence having several unrelated and often incorrect topic assignments. Of particular importance to us, the sentiment words are often assigned the wrong topics. Although not perfect, the results of running SentPOSLDA at the sentence level are more on-the-mark and more consistent within a sentence or sentence fragment discussing the same topics. We therefore run all of our experiments with this data set at the sentence level.



| |
|---|
| Great Hotel on Edge of French Quarter. I just spent 5 nights here during a convention. Very nice, large rooms. Very clean rooms and comfortable beds. Friendly, helpful staff. My first room was on the 4th floor and had a wonderful view of the roofs of the French Quarter but overlooked Bourbon Street and was a little noisy for me (even though the windows are double paned)--it's not that loud but I wasn't staying up late. They moved me for the next 4 nights to another very nice and quiet room on the inner courtyard. Nice, accommodating staff. I would stay here again. It's a little hike (10-15 minutes) to the convention center but I enjoyed the walk. You can take the Canal St. trolley but the trolley is SLOW and stops a few blocks from the convention center. The hotel is convenient to all the fun in the French Quarter. |
| Great Hotel on Edge of French Quarter. I just spent 5 nights here during a convention. Very nice, large rooms. Very clean rooms and comfortable beds. Friendly, helpful staff. My first room was on the 4th floor and had a wonderful view of the roofs of the French Quarter but overlooked Bourbon Street and was a little noisy for me (even though the windows are double paned)--it's not that loud but I wasn't staying up late. They moved me for the next 4 nights to another very nice and quiet room on the inner courtyard. Nice, accommodating staff. I would stay here again. It's a little hike (10-15 minutes) to the convention center but I enjoyed the walk. You can take the Canal St. trolley but the trolley is SLOW and stops a few blocks from the convention center. The hotel is convenient to all the fun in the French Quarter. |

| | | |
|---|---|---|
| Value | Front Desk | Positive |
| Room | Service | Negative |
| Location | Business Service | |
| Cleanliness | | |

**Figure 4.2:** Sample review tagged with topic and sentiment assignments. Upper: tagged by sentence-level SentPOSLDA, Lower: tagged by document-level SentPOSLDA

For a stronger qualitative analysis of the results, Table 4.7 shows top 30 aspect words for the 7 aspects learned from this data set and positive and negative words used to describe those aspects. Aspect cohesiveness is clear, as we have come to expect since the experiments with POSLDA have previously produced good results in this regard. Of more importance to us is the polarity and appropriateness of the sentiment words used to describe these aspects. As we can see, the relevance of the sentiment words to their assigned aspects is quite clear.

Turning to attention to quantitative analysis, in Table 4.8 we show the accuracy, precision and recall of SentPOSLDA on the TA1 data set. All three metrics show a very high level of performance.

| Aspect | Aspect words | Positive words | Negative words |
|---|---|---|---|
| Value | hotel, stayed, resort, price, nights, time, trip, place, experience, reviews, hotels, best, all, value, new, vacation, first, star, week, other, pay, money, husband, choice, returned, days, punta, cana, last, weekend | great, good, worth, nice, wonderful, excellent, beautiful, well, fantastic, perfect, money, amazing, happy, lovely, quality, fun, special, fabulous, pleasant, pretty, paradise, fine, positive, recommended, awesome, superb, cheap, expensive, kind, charming | bad, expensive, star, negative, worth, poor, horrible, terrible, rated, awful, inclusive, mexico, cancun, sad, hate, mean, bargain, des, st, riviera, ill, nervous, en, jamaica, accurate, worried, maya, playa, stars, mayan |
| Room | room, stay, night, rooms, view, floor, small, bathroom, bed, street, large, hotel, shower, beds, one, all, noise, it, square, suite, size, air, us, door, little, book, read, other, two, big | comfortable, nice, great, well, quiet, good, spacious, modern, tv, lovely, beautiful, fine, wonderful, excellent, water, pretty, fantastic, perfect, screen, bath, appointed, amazing, flat, fresh, separate, pleasant, bed, happy, comfy, equipped | old, bad, broken, tiny, dark, negative, worn, walls, poor, tired, horrible, terrible, stained, dated, stains, thin, smoking, awful, carpet, cold, loud, filthy, mold, like, mean, covered, uncomfortable, tiles, water, shabby |
| Location | hotel, location, beach, walk, restaurant, close, airport, area, city, station, all, minutes, bus, outside, minute, shopping, distance, restaurants, near, park, central, s, car, located, right, convenient, stop, short, train, center | great, good, nice, excellent, easy, walking, well, within, perfect, fantastic, beautiful, restaurants, wonderful, lovely, pretty, ideal, shop, amazing, fine, walked, fun, superb, near, fabulous, pleasant, awesome, wall, gorgeous, attractions, chinatown | located, bad, near, minutes, de, outside, min, minute, mins, away, blocks, less, steps, gracia, conveniently, steep, negative, passeig, scary, universal, st, euro, longer, narrow, studios, euros, poor, la, dead, residential |
| Cleanliness | people, kids, other, children, old, them, place, entertainment, music, all, time, smell, shows, valet, year, us, day, show, activities, casino, guests, group, lot, little, husband, parking, family, thing, son, smoking | clean, fun, great, good, well, nice, pretty, playing, volleyball, happy, linen, shoes, wonderful, games, funny, fine, dance, amazing, beautiful, lessons, excellent, fantastic, cleanliness, kind, played, perfect, special, salsa, lively, ping | dirty, bad, poor, horrible, moldy, mean, awful, trash, boring, ill, terrible, upset, repellant, unpleasant, urine, nasty, dead, sad, gross, unhappy, stupid, smelly, dreary, disgusting, foul, scary, pushing, horrendous, obnoxious, filthy |
| Front Desk | staff, helpful, us, hotel, front, desk, check, me, day, time, reception, wait, english, help, concierge, them, leave, people, morning, charge, manager, arrived, next, problem, flight, guest, reservation, way, one, other | friendly, nice, great, good, well, pleasant, polite, courteous, excellent, efficient, wonderful, happy, special, kind, spanish, spoke, easy, directions, fantastic, lovely, information, recommendations, desk, willing, fine, perfect, fun, accomodating, knowledgeable, pretty | rude, bad, poor, unfriendly, unhelpful, broken, terrible, beyond, mr, above, negative, upset, sad, delayed, indifferent, com, mean, cancelled, surly, awful, locked, unhappy, anxious, unpleasant, horrible, ill, angry, arrogant, pain, useless |
| Service | service, breakfast, food, pool, bar, lobby, free, buffet, day, all, coffee, restaurants, dinner, drinks, club, eat, area, drink, morning, best, go, property, it, lunch, lounge, luggage, one, time, wine, little | good, great, nice, excellent, well, fruit, fresh, wonderful, pretty, beautiful, lovely, eggs, breakfast, fine, amazing, fantastic, bread, fun, juice, special, cereal, pastries, toast, happy, fabulous, cereals, bacon, awesome, kind, bagels | free, bad, water, poor, beer, stocked, horrible, terrible, mini, bottled, soda, cold, awful, daily, bottles, mean, beers, soft, coke, drink, boring, hit, miss, red, negative, overpriced, upset, disgusting, pop, replenished |
| Business Service | hotel, san, business, access, internet, rooms, rate, francisco, parking, it, juan, place, hotels, old, deal, card, use, inn, booked, me, all, system, area, wireless, travel, company, high, paid, fee, lot | great, good, internet, nice, well, excellent, machine, beautiful, pretty, connection, wonderful, computer, lovely, wifi, fine, special, fantastic, perfect, amazing, pleasant, wi, happy, easy, fabulous, fun, charming, elegant, printer, massage, equipped | bad, old, cheaper, cool, trendy, poor, tired, online, terrible, misleading, mean, horrible, negative, tight, worn, accurate, awful, needed, certain, hate, exclusive, innovative, scary, disgusting, lacking, commercial, stepper, operational, metered, affordable |

**Table 4.7:** Top 30 aspect, positive and negative words discovered by SentPOSLDA in TA.

| Aspect | Accuracy | Precision | Recall |
|---|---|---|---|
| Overall | 0.856 | 0.873 | 0.968 |
| Value | 0.836 | 0.873 | 0.942 |
| Rooms | 0.863 | 0.899 | 0.945 |
| Location | 0.864 | 0.949 | 0.903 |
| Cleanliness | 0.894 | 0.936 | 0.949 |
| Front Desk | 0.973 | 0.912 | 0.946 |
| Service | 0.846 | 0.889 | 0.939 |
| Business Service | 0.809 | 0.849 | 0.937 |

**Table 4.8:** Quantitative results with the TA1 data set.

Looking at these results, it seems prudent to explain why the accuracy is lower than precision and recall. If we look at the raw results in Table 4.9 we see that when our model predicted a positive sentiment it was correct most of the time, whereas when it predicted a negative sentiment it did somewhat poorly. The performance of the negative predictions has brought the accuracy down while having little effect on precision and recall. The imbalance in performance with imbalanced data sets is to be expected as the bias will sway the outcome towards the dominant category.

| Aspect | True Positive | False Positive | False Negative | True Negative |
|---|---:|---:|---:|---:|
| Overall | 139,315 | 20,255 | 4,612 | 8,654 |
| Value | 101,259 | 14,717 | 6,192 | 5,699 |
| Rooms | 103,851 | 11,611 | 6,065 | 7,945 |
| Location | 87,199 | 4,709 | 9,351 | 1,834 |
| Cleanliness | 115,272 | 7,878 | 6,335 | 4,887 |
| Front Desk | 81,967 | 7,896 | 4,695 | 4,463 |
| Service | 104,395 | 12,985 | 6,811 | 4,494 |
| Business Service | 45,019 | 7,981 | 3,042 | 1,907 |

**Table 4.9:** Raw quantitative results for TA1 data set.

We next ran our sampler on the TA2 data set in order to compare it to Zhou's [51] results. For this experiment we used the same seed words that he used for the 5 aspects rated in his data set. We also added our seeds for front-desk and business service aspects into the seed list for the service aspect since they are closely related. Table 4.10 shows the results.

| | **Accuracy** | | **Precision** | | **Recall** | |
|---|---|---|---|---|---|---|
| **Aspect** | Zhou | SentPOSLDA | Zhou | SentPOSLDA | Zhou | SentPOSLDA |
| Overall | **0.868** | 0.826 | **0.908** | 0.830 | 0.634 | **0.970** |
| Service | 0.804 | **0.821** | 0.759 | **0.854** | 0.170 | **0.928** |
| Value | **0.848** | 0.732 | **0.865** | 0.771 | 0.608 | **0.908** |
| Rooms | **0.844** | 0.820 | **0.884** | 0.836 | 0.641 | **0.944** |
| Cleanliness | 0.793 | **0.848** | 0.890 | **0.900** | 0.499 | **0.936** |
| Location | 0.827 | **0.856** | 0.829 | **0.955** | 0.611 | **0.889** |

**Table 4.10:** Performance of SentPOSLDA in comparison to Zhou's method.

Overall the accuracy and precision scores are a little lower for TA2 than for TA1 and the recall is a little higher. Even though TA1 and TA2 come from the same source they do have different properties as explained in section 4.1 so some differences in the results are to be expected.

The accuracy and precision results in comparison to Zhou's method are somewhat mixed but in general an improvement. SentPOSLDA achieved better accuracy on 3 aspects and worse on 2. Similarily, SentPOSLDA achieved better precision scores on 3 aspects and worse on 2. It is also worth noting that when SentPOSLDA did worse it was by at most 0.09 points whereas when it did better it beat Zhou by as much as 0.12. The accuracy and precision for the overall rating was worse, but this is not surprising since our method uses an estimate instead of performing true overall sentiment analysis. The recall results, however, show a far superior performance by SentPOSLDA with most of the results being over 0.9 while Zhou's scores are all below 0.65 with one being as low as 0.17.

### 4.2.2 Reviews with Multiple Levels of Granularity

We performed two experiments with the Amazon data set. The first one was focused on the 7 overall topics (the product types) and the sentiments expressed about them. We set the number of topics to 8 (with the extra one for off-topic content) and ran the sampler at the document level. We did not provide any topic seed words. Table 4.11 shows the top 30 words for 7 of the learned topics and the positive and negative words used to describe those topics. Topic cohesiveness and the appropriateness of the sentiment words is still quite clear although we do see some non-sentiment words creeping into the list of detected sentiment words. Interestingly, the space heater and air conditioner topics ended up in the same category and SentPOSLDA identified an extra high-level topic that seems to correspond to Amazon's customer service.

In the second experiment we attempted to detect the more fine grained aspects of each type of product and the sentiments expressed towards them. Jo and Oh [22] set the number of aspects to 30 when performing this type of experiment so we did the same. Since each review talks about several aspects we ran this experiment at the sentence level.

There were a few aspects in our results that were not discernible but most of them were quite clear. Table 4.12 shows 5 aspects discovered for the laptop topic and the sentiment words associated with them. The sentiment words are still quite appropriate and more or less the same quality as in the document-level experiment.

| Topic | Topic Words | Positive Words | Negative Words |
|---|---|---|---|
| Canister Vacuum | vacuum, suction, use, carpet, dirt, canister, brush, dust, cord, small, get, little, hose, cleaning, filter, vac, hoover, floor, cleaner, one, house, handle, time, attachments, job, old, power, light, machine, years | great, well, easy, good, clean, floors, power, head, floor, attachment, nice, suction, canister, upright, pretty, carpet, carpets, happy, hard, wand, fine, excellent, miele, hardwood, rugs, furniture, perfect, vacuums, amazing, corners | hair, pet, bad, bissell, dog, dirty, eraser, pain, hate, messy, cats, poor, full, negative, roller, clogged, wet, haired, huge, junk, clogs, mean, belts, useless, heavy, black, place, belt, debris, broken |
| Space Heater / Air Conditioner | heater, unit, room, heat, use, fan, small, little, air, temperature, turn, hot, low, house, one, get, keep, set, cold, degrees, quiet, other, thermostat, time, high, bedroom, area, night, setting, product | heat, well, great, good, warm, heaters, room, space, heating, nice, easy, pretty, heats, setting, thermostat, living, bill, oil, fine, cold, temp, happy, ceramic, perfect, temperature, winter, off, effective, efficient, have | air, window, water, hose, cool, tank, cooling, bad, conditioner, drain, portable, c, propane, ac, exhaust, hole, cooler, box, ice, humidity, end, tube, btu, pull, heavy, full, mr, buddy, dry, block |
| Laptop | laptop, screen, computer, windows, use, keyboard, get, machine, drive, life, battery, hard, price, time, fast, software, vista, other, power, little, better, work, light, video, laptops, notebook, bit, ram, system, do | great, good, well, battery, nice, pretty, easy, laptop, fine, core, size, hp, acer, perfect, excellent, happy, dvd, wireless, hdmi, hd, home, mouse, performance, processor, amazing, small, beautiful, toshiba, awesome, resolution | macbook, pro, mac, apple, new, os, bad, pc, x, firewire, more, better, aluminum, glossy, leopard, display, macs, mbp, previous, trackpad, unibody, air, mean, solid, glass, matte, less, faster, user, macbooks |
| Coffee Machine | coffee, maker, water, machine, hot, cup, pot, carafe, use, brew, make, one, filter, time, get, coffeemaker, brewing, years, cuisinart, counter, pour, unit, lid, cups, other, little, grinder, buy, morning, better | great, good, cup, easy, clean, well, k, nice, cups, perfect, fresh, best, tea, fine, keurig, drink, home, excellent, happy, size, single, regular, espresso, own, pretty, starbucks, pods, different, travel, milk | water, basket, filter, bad, thermal, difficult, grounds, lid, top, plastic, brewing, pouring, place, design, hard, cone, negative, carafe, grinder, parts, level, braun, pain, plate, indicator, designed, krups, poor, pour, area |
| MP3 Player | ipod, player, music, use, video, sound, get, screen, zune, device, itunes, apple, quality, battery, better, play, software, new, touch, songs, thing, more, zen, nano, computer, other, product, go, buy, time | great, good, easy, well, nano, nice, ipod, songs, music, pretty, battery, shuffle, fm, gb, life, little, perfect, charge, excellent, happy, headphones, fine, games, ear, amazing, radio, cd, generation, awesome, mini | zune, software, media, bad, files, player, archos, hd, sync, fi, microsoft, firmware, support, x, looking, web, rhapsody, marketplace, wireless, wifi, folder, poor, pain, tracks, wi, internet, hate, interface, slow, digital |
| Digital SLR | camera, canon, lens, use, quality, lenses, get, nikon, cameras, flash, more, better, shoot, shots, pictures, dslr, light, time, s, mode, take, price, body, features, shooting, image, focus, picture, photos, auto | great, good, digital, lens, well, slr, pictures, easy, rebel, point, excellent, nice, amazing, film, happy, nikon, take, kit, pretty, battery, card, old, first, perfect, xt, photos, learn, fine, shoot, zoom | iso, noise, focus, sensor, image, bad, system, dust, white, bright, balance, lcd, performance, exposure, menu, frame, color, s, mark, high, less, panasonic, poor, dark, custom, view, metering, improved, viewfinder, low |
| Customer Service | product, one, new, problem, buy, first, time, unit, months, get, year, days, work, day, same, s, other, years, purchase, amazon, thing, box, do, company, me, problems, warranty, service, money, review | good, amazon, well, great, new, one, happy, shipping, return, same, fine, reviews, model, store, time, replacement, give, easy, price, brand, nice, excellent, com, few, year, pretty, kind, two, product, perfect | service, customer, support, warranty, repair, bad, told, call, send, hp, sony, poor, tech, center, few, number, broken, working, dead, called, junk, negative, fixed, phone, terrible, toshiba, repairs, email, horrible, rep |

**Table 4.11:** Top 30 topic, positive and negative words discovered by SentPOSLDA on the Amazon data set.

| **Aspect** | **Topic Words** | **Positive Words** | **Negative Words** |
|---|---|---|---|
| Hardware | ram, drive, processor, core, gb, memory, fast, faster, graphics, cpu, has, model, computer, performance, dual, up, card, intel, speed, hd, windows, have, machine, better, duo, ray, upgrade, blu, bit, ghz | hard, great, good, intel, more, enough, slow, well, gb, pretty, nice, ram, memory, single, easy, need, less, capable, know, uses, centrino, fine, amazing, reliable, awesome, running, laptop, excellent, celeron, efficient | slots, bad, nvidia, adequate, mean, x, gma, downgrade, powered, higher, versus, driver, controller, asus, knows, fingers, megabytes, h, ras, restored, heavily, fail, combined, kb, lags, int, crap, gigahertz, extreme, mahjong |
| Software | windows, software, mac, vista, os, system, use, run, xp, pc, laptop, installed, computer, programs, x, have, running, upgrade, operating, work, office, bit, all, comes, version, machine, free, problems, time, program | well, great, easy, install, good, better, fast, get, virus, new, fine, pro, nice, microsoft, work, clean, pretty, friendly, upgraded, restore, different, awesome, fresh, happy, latest, boot, spent, intuitive, amazing, trial | bad, come, junk, open, crap, hate, pain, crappy, mean, poor, negative, malicious, unwanted, burn, efficiently, everything, awful, aol, stupid, worthless, increase, universal, email, symantec, sad, garbage, dead, terrible, restoration, spyware |
| Gaming Performance | games, graphics, laptop, play, video, card, high, all, expectations, it, run, game, computer, gaming, runs, machine, settings, fast, performance, do, nvidia, playing, better, everything, end, running, resolution, exceeded, integrated, get | well, great, good, pretty, fine, use, nice, call, discrete, awesome, full, amazing, fun, expect, turbo, happy, kind, low, perfect, nvidia, uses, easy, ones, beautiful, unreal, stunning, forget, fantastic, excellent, impressive | dead, maze, bad, developer, improvement, evil, shader, r, ferrari, spore, conserve, google, junk, l, jedi, deluxe, titles, horsepower, released, fiberoptical, claims, k, pinball, resolving, pseudo, alert, klondike, command, graohics, camcorder |
| Wi-Fi Performance | wireless, movies, video, watching, watch, videos, web, tv, screen, wifi, use, music, laptop, play, get, internet, computer, hd, youtube, no, games, n, network, connection, home, have, browser, router, shows, built | great, well, internet, good, movie, easy, connect, browsing, fast, nice, fine, using, pretty, awesome, excellent, email, fi, lot, work, streaming, running, fun, amazing, perfect, happy, systems, range, d, kind, share | negative, crashing, poor, bad, regularly, choppy, text, index, launching, optimal, manages, consider, start, application, decision, sub, fail, correctly, hl, boring, example, conditions, brother, elements, met, ubuntu, introduced, interest, nas, spike |
| Keyboard and Mouse / Touch Pad | keyboard, it, touch, pad, button, screen, use, up, mouse, keys, buttons, touchpad, click, right, like, little, trackpad, finger, key, used, back, laptop, bit, feel, find, left, wheel, type, fingers, typing | nice, easy, get, great, hard, good, well, down, pretty, multi, rest, fine, fn, using, touch, side, clean, kind, end, lock, excellent, black, beautiful, awesome, scrolling, around, perfect, glass, amazing, delete | bad, smooth, trackpad, pain, mean, terrible, located, loose, hate, negative, horrible, place, awful, poor, touches, surfaces, fat, lightly, pattern, clumsy, rattle, fail, factor, brush, ugly, disaster, overly, damage, resulting, crap |

**Table 4.12:** Top 30 aspect, positive and negative words detected by SentPOSLDA for the Laptop topic in the Amazon data set.

The quantitative results are shown in Table 4.13. Overall, SentPOSLDA performed slightly better when allowed to focus on fine-grained aspects. This data set has different properties and contains reviews about different topics than the TripAdvisor data set so some differences in the results are to be expected. Nevertheless, we suspect one of the factors for worse performance is the fact that this is a more complex data set with multiple levels of granularity and a much larger number of fine-grain aspects.

| Granularity | Accuracy | Precision | Recall |
|---|---|---|---|
| Document level, 7 topics | 0.779 | 0.779 | 0.999 |
| Sentence level, 30 aspects | 0.780 | 0.783 | 0.993 |

**Table 4.13:** Quantitative results on the Amazon data set.

### 4.2.3 Noisy Data

Twitter posts present the greatest challenge to any kind of NLP task for several reasons. Twitter's limit of 140 characters per post can force the user to abbreviate words or compromise on grammar. The posts are also riddled with emoticons, hashtags, overused exclamation marks and other noise. Furthermore, the short length of these posts do not provide much information for us to go on. The amount of topics discussed on Twitter is also a challenge since Twitter is open to everyone to talk about anything. The SemEval data set that we used is manually marked for topics and we counted 602 of them. Of the three types of data sets, SentPOSLDA performed the worst on this one as the scores in Table 4.14 show. Similarly to POSLDA, qualitative analysis revealed few discernible topics and sentiments. The ones we did find are presented in Table 4.15.

| Metric | Score |
|---|---|
| Accuracy | 0.713 |
| Precision | 0.716 |
| Recall | 0.988 |

**Table 4.14:** Quantitative results with the Twitter data set.

| **Topic Words** | **Positive Words** | **Negative Words** | **Comments** |
|---|---|---|---|
| Asia, economic, office, growth, china, ost, region, forecast, right, trails | rises, lag, upbeat | | Positive reaction to news about Asian economy. |
| capello, russia, fabio, portugal, match, intended, said, manager, result, cantwait | good, wishes | honesty, aloud | Comments about a sports match. |
| monument, washington, south, korea, born, seoul, solo, travel, ur, continued | beautiful, country, gorgeous, society, japan, stunning | | Comments about traveling through Asian countries. |
| cloud, expo, computing, ceo, track, data, wppi, power, ab, speed | effective, great | caught | Reviews about new technologies, possibly presented at a conference. |
| garden, olive, social, dinner, series, deeds, week, hosting, perry, tyler | happy, good, ikayrob, pretty, antones, cute, special | us, terrible | Review about a dinner at a restaurant. |
| tournament, kickball, your, hands, match, soccer, arm, ddliverpool, octobr, ipittythefoolthatcantkick, itoldschool | happy, drive, winning | s***, hate | Sentiments about a soccer game. |
| goal, zone, field, nhllockout, preds, kept, growing, fix, jonesing, holes | awesome, great, territory | bad, pain | Sentiments about a football game. |
| lead, pacers, pistons, his, picks, cold, bench, bulls, trail, pts | reclaim, audition | foul, upset | Sentiments about another football game. |
| black, camera, deals, olympus, jay, case, om, bag, digital, lens | happy, good, favorite, great, pinaghuhugutan, freeblog, yelawolf, shaped, kickback | oled | Comments about a camera. |
| million, ordered, ipad, shipped, child, cow, mini, suggest, released, bethere | friendly, well | s*** | Comments about a new Apple product. |
| yay, nominated, woohoo, grammy, sweetheart, nominations, considering, joan, jasmine, routes | happy, kul | omfg | Sentiments about a Grammy nomination. |
| trailer, games, by, bioshock, infinite, playstation, mashasha, dungeon, february, gen | great, controversy, previous, awesome, squeeing, kindly | | Positive comments about a recent video game release. |

**Table 4.15:** Topic and sentiment words discovered by SentPOSLDA on the Twitter data set.

## 4.2.4 Effects of Topic Filtering

One of the fine-tuning features added to SentPOSLDA is the filtering of topics for the sentiment words. As explained in Chapter 3, when building a model from which to draw the word sentiment we ignore all topics that were not mentioned in the sentence. To test the effect this has on the results we ran SentPOSLDA against the TA1 data set with and then without topic filtering. The results are shown in Table 4.16 and clearly show an improvement.

| | **Accuracy** | | **Precision** | | **Recall** | |
|---|---|---|---|---|---|---|
| **Aspect** | With | Without | With | Without | With | Without |
| Overall | **0.856** | 0.852 | 0.873 | 0.879 | **0.968** | 0.953 |
| Value | **0.836** | 0.798 | 0.873 | 0.879 | **0.942** | 0.881 |
| Room | **0.863** | 0.848 | 0.899 | 0.918 | **0.945** | 0.902 |
| Location | **0.864** | 0.793 | 0.949 | 0.945 | **0.903** | 0.828 |
| Cleanliness | **0.894** | 0.852 | 0.936 | 0.939 | **0.949** | 0.896 |
| Front Desk | **0.973** | 0.845 | 0.912 | 0.925 | **0.946** | 0.895 |
| Service | **0.846** | 0.803 | 0.889 | 0.882 | **0.939** | 0.892 |
| Business Service | **0.809** | 0.769 | 0.849 | 0.858 | **0.937** | 0.865 |

**Table 4.16:** Quantitative results with and without topic filtering on the TA1 data set.

## 4.2.5 Effects of Aspect Rating Approximation

An additional fine-tuning heuristic in our solution is the approximation of the aspect rating using the reviewer's overall rating rather than allowing equation (3.17) to reduce to the overall sentiment of the corpus. We expect this feature to have more impact the less bias there is in the data set since in a heavily biased corpus the overall corpus sentiment will be the correct one most of the time.

Indeed, as Figure 4.17 shows there is virtually no effect on the TA1 data set which is very heavily biased towards the positive sentiment. In Figure 4.18, we see a greater effect on the TA2 data set which is less biased.

| | **Accuracy** | | **Precision** | | **Recall** | |
|---|---|---|---|---|---|---|
| **Aspect** | With | Without | With | Without | With | Without |
| Overall | 0.856 | 0.847 | 0.873 | 0.847 | 0.968 | 0.997 |
| Value | 0.836 | 0.817 | 0.873 | 0.864 | 0.942 | 0.929 |
| Room | 0.863 | 0.873 | 0.899 | 0.905 | 0.945 | 0.949 |
| Location | 0.864 | 0.861 | 0.949 | 0.943 | 0.903 | 0.907 |
| Cleanliness | 0.894 | 0.906 | 0.936 | 0.933 | 0.949 | 0.966 |
| Front Desk | 0.973 | 0.879 | 0.912 | 0.913 | 0.946 | 0.954 |
| Service | 0.846 | 0.840 | 0.899 | 0.877 | 0.939 | 0.947 |
| Business Service | 0.809 | 0.804 | 0.849 | 0.840 | 0.937 | 0.944 |

**Table 4.17:** Quantitative results with and without aspect rating approximation on the TA1 data set.

| | **Accuracy** | | **Precision** | | **Recall** | |
|---|---|---|---|---|---|---|
| **Aspect** | With | Without | With | Without | With | Without |
| Overall | **0.826** | 0.800 | **0.830** | 0.794 | **0.970** | 0.995 |
| Service | 0.821 | 0.819 | 0.854 | 0.854 | 0.928 | 0.923 |
| Value | **0.732** | 0.700 | **0.771** | 0.753 | **0.908** | 0.887 |
| Rooms | 0.820 | 0.824 | 0.836 | 0.841 | 0.944 | 0.944 |
| Cleanliness | 0.848 | 0.846 | **0.900** | 0.876 | 0.936 | **0.952** |
| Location | 0.856 | 0.863 | 0.955 | 0.952 | 0.889 | **0.899** |

**Table 4.18:** Quantitative results with and without aspect rating approximation on the TA2 data set.

## 4.3 Parallel Gibbs Sampling Performance

For all the experiments in section 4.2 we have used the parallel implementation of our Gibbs sampling algorithm. We wanted to be sure that the parallel implementation did not degrade the results of our experiments. To that end we measured the performance of SentPOSLDA starting with a single thread and then doubling the number of threads all the way to 8129 threads – the maximum supported by our GPU. For time considerations we used a randomly sampled subset of the TA1 data set that was 10% the size of the original. As Figures 4.3. and 4.4 show, the accuracy, precision and recall of the results stayed quite constant through all of these experiments while the execution time roughly halved every time we doubled the number of threads. Most notably, the execution time went from 33 days with a single thread to just 40 minutes with the maximum number of threads.

For comparison purposes, we have included the execution time of the single-threaded sampler on our CPU. The CPU has a higher per-core frequency – 2.1 GHz as opposed to the GPU's 850 MHz – and certain optimizations that are not present in the GPU. As a result, the CPU was faster than the GPU running at 64 threads or less but these advantages crumbled as we increased the number of threads.

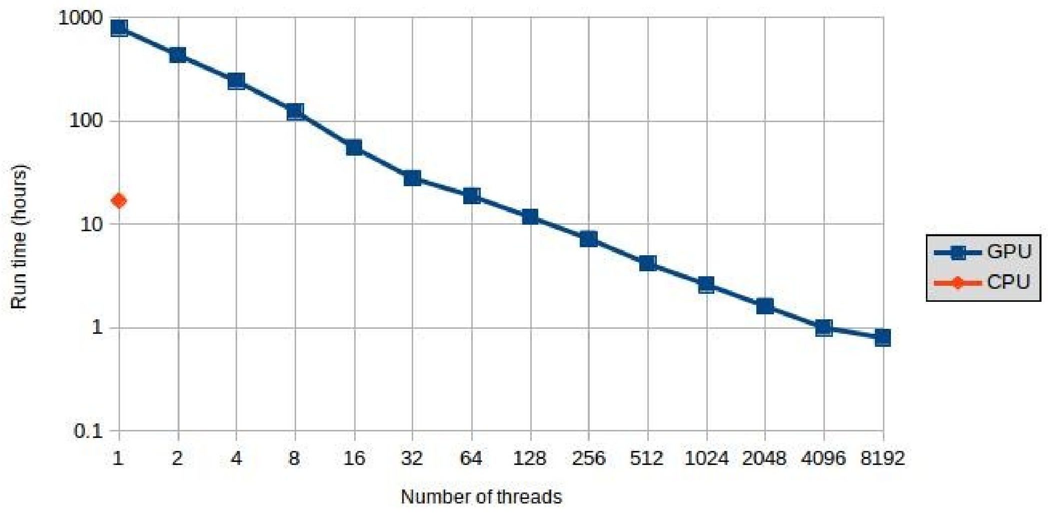


**Figure 4.3:** Run time as the number of threads varies.

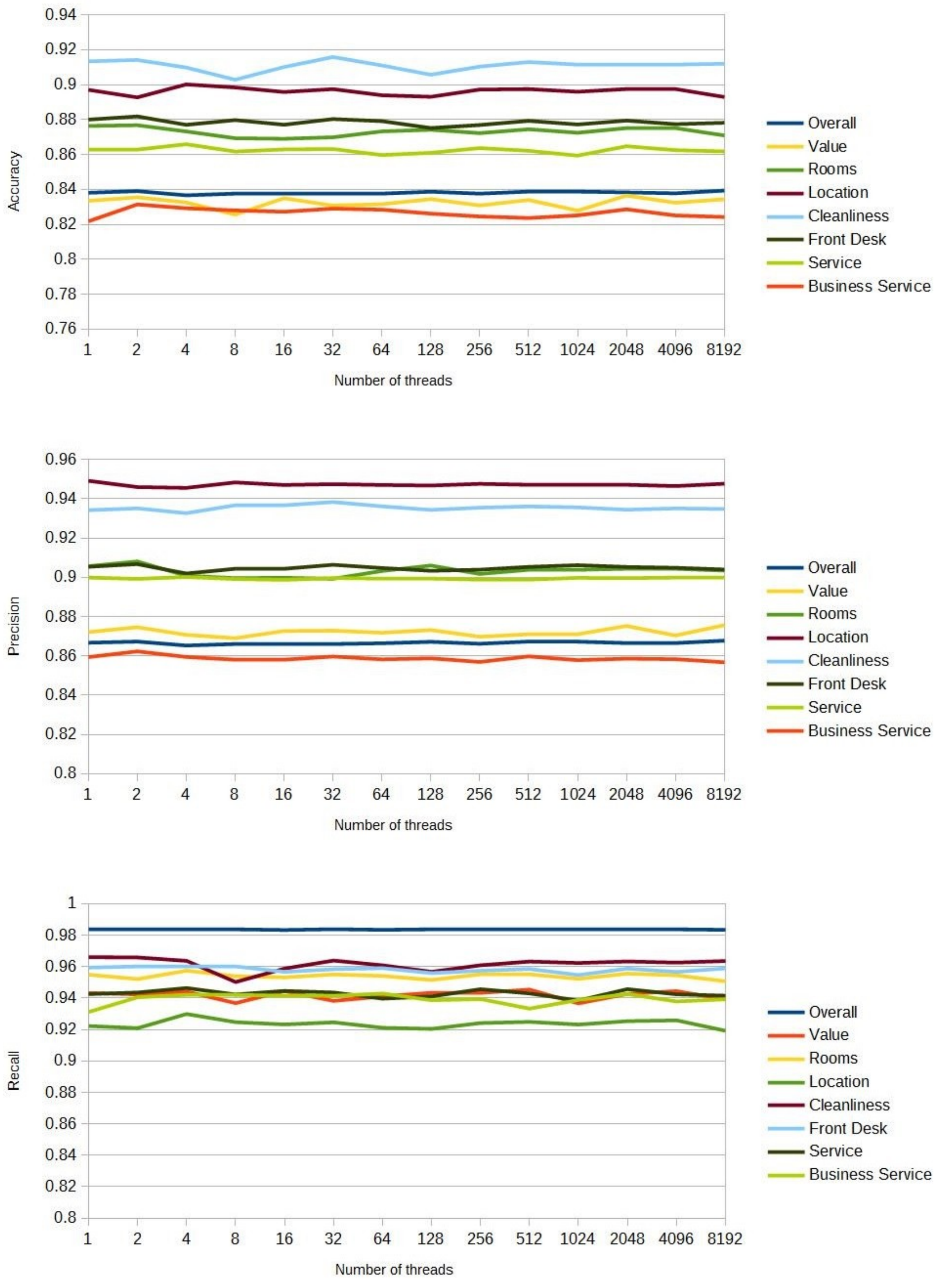


**Figure 4.4:** Accuracy, precision and recall as the number of thread varies.

# Chapter 5

# Conclusions and Future Work

## 5.1 Conclusions

We introduced a new probabilistic model for aspect-level sentiment analysis and a multi-threaded implementation of the Gibbs sampling algorithm for faster inference. Our model is largely based on the POSLDA [9] model – a topic classifier that incorporates syntax modelling in order to separate semantic words from purely functional words. Within semantic words, we model the probability of a word expressing sentiment using the ideas presented in STDP [24]. Unlike STDP, we do not need a POS tagger since our model includes a syntax modelling component. We perform sentiment analysis using a sentiment modelling component similar to JST [25] but restricting it to the words in semantic classes where the presence of sentiment was identified. Based on the desired output, we restructure the sentiment modelling component so it has a sentiment distribution for each topic as opposed to a topic distribution for each sentiment as in JST, ASUM [22] and STDP. Instead of modelling overall sentiment explicitly, we approximate it using aspect sentiments and topic distributions. Finally, we add a few heuristics such as topic filtering and aspect rating approximation to get the best performance out of the resulting model.

We performed experiments on three different types of data. The TripAdvior data set contains reviews on hotels and provides ratings on 7 aspects giving this data set a single level of granularity. We used aspect seed words to force the discovered aspects into the 7 predetermined ones so we we can compare the output with the given ratings. Our model achieved a very high level of performance in this setting.

The second data set is a set of reviews on 7 different types of products from the Amazon web site. Each of these product types has several fine-grained aspects which means that this data set has two levels of granularity. The aspects were not given to us a priori and the provided ratings were on overall satisfaction only. The performance of our model suffered in this case although still achieving scores around 78%. One of the implications of the performance drop is that our model's ability to handle multiple levels of granularity is something that can be improved upon.

Finally, we tested our model against a set of Twitter posts. This data set presented many challenges including incorrect spelling and grammar, use of hashtags, emoticons and other noise, and lack of information due to the extremely short length of each post. Similarly to POSLDA, our model performed the worst with this data set.

We also compared the performance of our model with the approach taken by Zhou. Our approach demonstrated modest improvements in accuracy and precision and a drastic improvement in recall. The data set used for this experiment was very biased towards the positive. The fact that our model achieved much better recall suggests that it is more sensitive to the overall bias of the corpus.

We introduced the notion of topic filtering in order to assign the correct aspect to sentiment words. This is something that is lacking in all other models we reviewed in this thesis. Our approach was to simply restrict the available aspects to the ones mentioned in the current sentence using non-sentiment words. The results of our experiments showed that including this heuristic improved performance, suggesting that some type of target detection for sentiment words is necessary.

We identified a potential problem with applying a probability function for aspect level sentiment analysis to documents where some of the aspects were not mentioned. In such a case, the function reduces to the overall sentiment of the corpus. In a heavily biased corpus the consequence would be unnoticeable since the overall sentiment would be correct most of the time. However, with more balanced data sets the effect is stronger. To overcome this issue we calculate the overall sentiment while ignoring aspects that were not mentioned and then use the overall sentiment as an approximation for the sentiment of those aspects. Our experiments confirmed our suspicions with both heavily biased and more balanced data sets suggesting some heuristic considerations are appropriate with probabilistic models such as ours.

Finally, we measured the performance of our model as we changed the number of threads in order to examine the effects of multi-threading. The scores, as measured by all three metrics, remained roughly constant for all 7 aspects of the TripAdvisor data set as we increased the number of threads from 1 to 8192. Meanwhile, the execution time decreased from 33 days to just 40 minutes. These results demonstrate that the effects of multi-threading on the quality of results in negligible while the benefits are enormous.

## 5.2 Future Work

We spent a great deal of time talking about and managing different levels of granularity. Our approach has been to use fewer topics on a document-level SentPOSLDA when each document contains one topic and more topics on a sentence-level SentPOSLDA when each document discusses several topics or aspects. A much better approach would be to include granularity into the model itself. A glaring candidate for this would be to use Multi-Grain LDA (MG-LDA) [38] instead of LDA [7] as the basis for the topic modelling component. MG-LDA performs topic modelling at two levels: high level topics such as the Laptop, MP3 Player, Space Heater topics in the Amazon data set, and low-level aspects such as battery life, image quality, ease of use, etc. With topics and aspects structured in such a way the model would be able to detect the rateable aspects for each topic more successfully and automatically.

Darling's work on POSLDA [9] includes NP-POSLDA, a non-parametric version of the model that does not require the number of topics to be provided ahead of time. An interesting avenue of research would be to change the basis of SentPOSLDA from POSLDA to NP-POSLDA and test it on data sets such as the Amazon one where the number of aspects is unknown. The resulting model would be able to find the optimal number of topics which should result in better topic classification and therefore better performance on sentiment analysis.

One of our improvements over the other models was including a more accurate way to assign topics to sentiment words. Our approach was very simple – we simply let non-sentiment, semantic words in the current sentence dictate the available topics for the sentiment words. This heuristic alone resulted in improvement gains. However, it is entirely possible for a sentiment word to refer to something from another sentence. For example, consider the following excerpt:

“The room was great. It was very comfortable and spacious.”

The words “comfortable” and “spacious” are referring to the topic “room” mentioned in the previous sentence. Our approach would miss the sentiment expressed in the second sentence entirely. If a simple heuristic such as ours is showing promising results, a more sophisticated approach to finding the target of a sentiment word would surely produce even better results. A possible avenue of research would be to take advantage of the syntax information contained within our model.

SentPOSLDA does not model overall sentiment explicitly but rather approximates it using the aspect sentiments and topic distributions. This has proven effective in our experiments but proved to be much worse than Zhou's [51] method. Incorporating overall sentiment into the model itself could be a way to bring the performance of overall sentiment detection up to par.

Finally, our approach did not handle negation in any way, which was likely the reason for some of the incorrect sentiment assignments. Finding a method to handle negation should result in a further uptick in performance.

# Bibliography

[1] Amr Ahmed, Mohamed Aly, Joseph Gonzalez, Shravan Narayanamurthy and Alexander Smola. Scalable Inference in Latent Variable Models. In Proceedings of the fifth ACM international conference on Web search and data mining, pages 123-132, Seattle, WA, USA, 2012. ACM.

[2] Arthur Asuncion, Padhraic Smyth and Max Welling. Asynchronous Distributed Learning of Topic Models. In Advances in Neural Information Processing Systems 21 (NIPS 2008), pages 81-88, Vancouver, British Columbia, Canada, 2008. NIPS.

[3] Bin Bi, Yuanyuan Tian, Yannis Sismanis, Andrey Balmin and Junghoo Cho. Scalable topic-specific influence analysis on microblogs. In Proceedings of the 7th ACM international conference on Web search and data mining, pages 513-522, New York, NY, USA, 2014. ACM.

[4] Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer Science + Business Media, LLC, New York, NY, USA, 2006

[5] David M. Blei, Michael I. Jordan, Thomas L. Griffiths and Joshua B. Tenenbaum. Hierarchical Topic Models and the Nested Chinese Restaurant Process. Journal of the ACM: 1-30, 2010.

[6] David M. Blei and John D. Lafferty. Correlated Topic Models. Y. Weiss, B. Schölkopf, and J. Platt editors. In Advances in Neural Information Processing Systems 18, pages 147-154. MIT Press, Cambridge, MA, USA, 2006

[7] David M. Blei, Andrew Y. Ng, Michael I. Jordan. Latent Dirichlet Allocation. Journal of Machine Learning Research 3: 993-1022, 2003.

[8] Bob Carpenter. Integrating out Multinomial Parameters in Latent Dirichlet Allocation and Naive Bayes for Collapsed Gibbs Sampling. LingPipe Inc, 2010.

[9] Willam M. Darling. Generalized Probabilistic Topic and Syntax Models for Natural Language Processing. Ph.D. thesis, University of Guelph, 2012.

[10] Kushal Dave, Steve Lawrence and David M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In Proceedings of the 12th international conference on World Wide Web, pages 512-528, Budapest, Hungary, 2003. ACM.

[11] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer and Richard Harshman. Indexing by latent semantic analysis. Journal of the American Society for Information Science: 41: 391–407, 1990.

[12] Xiaowen Ding, Bing Liu and Philip S. Yu. A Holistic Lexicon-Based Approach to Opinion Mining. In Proceedings of the 2008 International Conference on Web Search and Data Mining, pages 231-240, Palo Alto, CA, USA, 2008. ACM.

[13] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan and Noah A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, pages 42–47, Portland, Oregon, 2011. Association for Computational Linguistics.

[14] Andrew B. Goldberg and Xiaojin Zhu. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In Textgraphs 2006 Workshop on Graph-based

Algorithms for Natural Language Processing, pages 45-52, Stroudsburg, PA, USA, 2006. The Association for Computational Linguistics.

[15] Barbara B. Greene and Gerald M. Rubin. Automatic grammatical tagging of English. Department of Linguistics, Brown University, Providence, Rhode Island, 1971

[16] Thomas L. Griths, Mark Steyvers, David M. Blei and Joshua B. Tenenbaum. Integrating topics and syntax. Advances in Neural Information Processing Systems, 17: 537-544, 2005.

[17] Vasileios Hatzivassiloglou and Kathleen R. McKeown. Predicting the Semantic Orientation of Adjectives. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, pages 174-181, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

[18] Vasileios Hatzivassiloglou and Janyce M. Wiebe. Effects of Adjective Orientation and Gradability on Sentence Subjectivity. In Proceedings of the 18th conference on Computational linguistics, pages 299-305, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[19] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. Machine Learning: 42(1-2):177-196, 2001.

[20] Minqing Hu and Bing Liu. Mining and Summarizing Customer Reviews. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 168-177, Seattle, WA, USA, 2004. ACM.

[21] Matthew Hurst and Kamal Nigam. Retrieving Topical Sentiments from Online Document Collections. In Proceedings of Document Recognition and Retrieval, pages 27-34, San Jose, CA, USA,

2004. SPIE.

[22] Yohan Jo and Alice 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, Hong Kong, China, 2011. ACM.

[23] Soo-Min Kim and Eduard Hovy. Determining the Sentiment of Opinions. In Proceedings of the 20th International Conference on Computational Linguistics, pages 1-7, Geneva, Switzerland, 2004. Association for Computational Linguistics.

[24] Chengtao Li, Jianwen Zhang, Jian-Tao Sun, and Zheng Chen. Sentiment Topic Model with Decomposed Prior. In SIAM International Conference on Data Mining (SDM'13), pages 767-775, Austin, TX, USA, 2013. Society for Industrial and Applied Mathematics.

[25] 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, Hong Kong, China, 2009. ACM.

[26] Zhiyuan Liu, Yuzhou Zhang, Edward Y. Chang and Maosong Sun. PLDA+: Parallel latent dirichlet allocation with data placement and pipeline processing. ACM Transactions on Intelligent Systems and Technology: 1-18, 2011.

[27] David Mimno and Andrew McCallum. Organizing the OCA: Learning Faceted Subjects from a Library of Digital Books. In Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries , pages 376-385, Vancouver, British Columbia, Canada, 2007. ACM.

[28] Thomas P. Minka. Estimating a Dirichlet Distribution. Annals of Physics: 1-13, 2003.

[29] David Newman, Arthur Asuncion, Padhraic Smyth and Max Welling. Distributed Algorithms for Topic Models. The Journal of Machine Learning Research: 1801-1828, 2009.

[30] David Newman, Padhraic Smyth and Mark Steyvers. Scalable Parallel Topic Models. Journal of Intelligence Community Research and Development: 1-13, 2006.

[31] Bo Pang and Lillian Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pages 115-124, Ann Arbor, MI, USA, 2005. Association for Computational Linguistics.

[32] Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing, pages 79-86, Philadelphia, PA, USA, 2002. Association for Computational Linguistics.

[33] Judea Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco, CA, USA, 1988

[34] Ellen Riloff, Janyce Wiebe and Theresa Wilson. Learning Subjective Nouns using Extraction Pattern Bootstrapping. In Proceedings of the Seventh Conference on Natural Language Learning, pages 25-32, Edmonton, Canada, 2003. Association for Computational Linguistics.

[35] Jieying She and Lei Chen. TOMOHA: TOpic model-based HAshtag recommendation on twitter. In Proceedings of the 23rd International Conference on World Wide Web, pages 371-372, Seoul, Korea, 2014. International World Wide Web Conferences Steering Committee.

[36] Liangcai Shu, Bo Long and Weiyi Meng. A Latent Topic Model for Complete Entity Resolution. In 25th IEEE International Conference on Data Engineering, pages 880-891, Shanghai, China, 2009. IEEE Computer Society.

[37] Khoat Than and Tu Bao Ho. Fully Sparse Topic Models. In Proceedings of the 2012 European conference on Machine Learning and Knowledge Discovery in Databases, pages 490-505, Berlin, Germany, 2012. Springer-Verlag.

[38] Ivan Titov and Ryan McDonald. Modeling Online Reviews with Multi-grain Topic Models. In Proceedings of the 17th international conference on World Wide Web, pages 111-120, Beijing, China, 2008. ACM.

[39] Richard M. Tong. An operational system for detecting and tracking opinions in on-line discussions. In Workshop on Operational Text Classification Systems, pages 1-6, New Orleans, LA, USA, 2001. ACM.

[40] Peter D. Turney. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In Proceedings of theTwelfth European Conference on Machine Learning, pages 491-502, Berlin, Germany, 2001. Springer-Verlag.

[41] 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, Philadelphia, PA, USA, 2002. Association for Computational Linguistics.

[42] Hanna M. Wallach. Structured Topic Models for Language. Ph.D. thesis, University of Cambridge, 2008.

[43] Yi Wang. Distributed Gibbs Sampling of Latent Dirichlet Allocation: The Gritty Details. Technical Report. 2007.

[44] Xiaogang Wang and Eric Grimson. Spatial Latent Dirichlet Allocation. J. C. Platt and D. Koller and Y. Singer and S. T. Roweis editors. In Advances in Neural Information Processing Systems 20, pages 1577-1584. Curran Associates, Inc., Red Hook, NY, USA, 2008

[45] Hongning Wang, Yue Lu and Chengxiang Zhai. Latent Aspect Rating Analysis on Review Text Data: A Rating Regression Approach. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 783-792, Washington, DC, USA, 2010. ACM.

[46] Hongning Wang, Yue Lu and ChengXiang Zhai. Latent Aspect Rating Analysis without Aspect Keyword Supervision. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 618-626, San Diego, CA, USA, 2011. ACM.

[47] Janyce M. Wiebe. Learning Subjective Adjectives from Corpora. In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pages 735-740, Menlo Park, CA, USA, 2000. American Association for Artificial Intelligence.

[48] Janyce Wiebe, Rebecca Brucey, Matthew Bell, Melanie Martinz and Theresa Wilson. A Corpus Study of Evaluative and Speculative Language. In Proceedings of the 2nd ACL SIGdial Workshop on Discourse and Dialogue, pages 1-10, Aalborg, Denmark, 2001. Association for Computational Linguistics.

[49] Jian-Feng Yan, Zhi-Qiang Liu, Yang Gao and Jia Zeng. Communication-efficient algorithms for parallel latent Dirichlet allocation. Soft Computing - A Fusion of Foundations, Methodologies and Applications: 3-11, 2015.

[50] Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu and Wayne Niblack. Sentiment Analyzer: Extracting Sentiments about a Given Topic using Natural Language Processing Techniques. In Proceedings of the Third IEEE International Conference on Data Mining, pages 427-434, Melbourne, FL, USA, 2003. IEEE.

[51] Haochen Zhou. Aspect-Level Sentiment Analysis Based on a Generalized Probabilistic Topic and Syntax Model. M.Sc. thesis, University of Guelph, 2014.

[52] Jun Zhu, Xun Zheng, Li Zhou and Bo Zhang. Scalable inference in max-margin topic models. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 964-972, Chicago, IL, USA, 2013. ACM.

# APPENDICES

## Appendix A: Seed Words for Sentiment

| **Positive words** | **Negative words** |
|---|---|
| adorable amazing angelic appealing attractive awesome beautiful beneficial bliss bravo brilliant charming cheery clean commend courageous cute dazzling delight delightful divine easy ecstatic effective efficient effortless electrifying elegant enchanting energetic engaging enthusiastic excellent exciting exquisite fabulous fantastic favorable fine flourishing fresh friendly fun funny generous glamorous glowing good gorgeous graceful great handsome happy harmonious healthy hearty heavenly honest ideal impressive jovial joy jubilant keen kind knowledgeable legendary lively lovely marvelous masterful miraculous nice nurturing nutritious paradise perfect phenomenal pleasurable plentiful pleasant polished positive pretty refined reliable remarkable respected robust satisfactory skilled skillful soulful sparkling special stupendous stunning superb terrific thrilling thriving tranquil truthful unreal unwavering upbeat upstanding valued vibrant virtuous well wonderful wondrous yummy zeal zealous | abysmal alarming angry annoy anxious appalling atrocious awful bad banal belligerent boring broken callous carelessly clumsy contradictory corrosive corrupt crap crappy creepy criminal cruel cry dead decaying damage damaging dastardly deplorable deprived deformed deny despicable detrimental dirty disgusting disheveled dishonest dishonorable dismal distress dreadful dreary enraged evil fail faulty fear feeble filthy foul frightful gawky ghastly greed grim grimace gross grotesque gruesome haggard harmful hate hideous homely horrendous horrible hostile hurt hurtful icky ignorant ill immature inane inelegant infernal insane insidious insipid jealous junk junky lousy malicious mean menacing messy misshapen moldy monstrous naive nasty naughty negative nonsense noxious objectionable odious offensive oppressive pain pessimistic petty poisonous poor prejudice questionable reject renege repellant repulsive repugnant revolting rocky rotten rude ruthless sad savage scare scary scream shit shitty shoddy shocking sickening sinister slimy smelly spiteful stinky stressful stupid substandard terrible terrifying threatening ugly undermine unfair unfavorable unhappy unhealthy unjust unlucky unpleasant unsatisfactory unsightly unwanted unwelcome unwholesome unwieldy unwise upset vicious vile villainous vindictive wicked woeful worthless wound yell yucky |

## Appendix B: Seed Words for TripAdvisor Data Set

| **Aspect** | **Seed Words** |
|---|---|
| Value | experience cheap quality cost expectation honeymoon vacation accommodate discount atmosphere anniversary expensive pricy accomodate accommodation accomodation dollar price bargain recommended star pay value choice resort money worth start overprice |
| Rooms | bathroom bed size tower window suite sleep decorate air decor bedroom furnish renovation suit apartment bathtub soap security television shampoo mirror hairdryer floor square shower space inside mansion channel furniture toiletry louver courtyard carpet house room quiet light balcony pillow Queen kitchen tv view night noise condition view stay comfortable street spacious book chair double king request read bath upgrade overlook housekeeping modern separate sink conditioner |
| Location | location shop station locate bus airport outside taxi site facility market tube train wall traffic museum bank underground transport pantheon transportation position boulevard touristy conference walk street shuttle boutique plaza opera supermarket tram park transport mall avenue close beach sight-see surround restaurant distance minute center metro bloc central min stop near convenient route |
| Cleanliness | clean dirty nonsmoking valet smoke linen smell tidy maintain smoker resort linen cleanliness musty cigarette spotlessly |
| Front Desk | staff reception wait concierge reservation receptionist checkout office management inform guide check charge request manager entrance owner desk bellman contact check-in front arrive book leave guest welcome courteous greet smile helpful friendly english help information polite waiter rude question |
| Service | breakfast service restaurant food park coffee cafe drink dinner buffet buffet tea towel lounge luggage club elevator gym fridge yogurt cake suitcase plasma movie kitchen toast meat supply newspaper security waiter laundry facility cheese bread snack wine croissant gem property egg juice swim lobby garage website belvedere bar serve smile pool meal fruit cereal speak lunch free seafood steak selection gym variety bacon |
| Business Service | business center computer management wifi company route massage facility website machine computer connection online system internet access wireless speed laptop fitness printer print high-speed wi-fi broadband pc ethernet network facial fax therapist pcs speedboat speedos lan password modem router cord connectivity hi-speed download computerize highspeed dial-up speedboats |