Topic Modelling for Short Text

by

Jocelyn Rangarirai Mazarura

Submitted in partial fulfilment of the requirements for the degree

Magister Scientiae in Mathematical Statistics

In the Department of Statistics

In the Faculty of Natural and Agricultural Sciences

University of Pretoria

30 April 2015

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

I, *Jocelyn Rangarirai Mazarura*, declare that this dissertation, which I hereby submit for the degree Magister Scientiae in Mathematical Statistics at the University of Pretoria, is my own work and has not previously been submitted by me for a degree at this or any other tertiary institution.

Signature: 

Date: 30/04/2015

ABSTRACT

Over the past few years, our increased ability to store large amounts of data, coupled with the increasing accessibility of the internet, has created massive stores of digital information. Consequently, it has become increasingly challenging to find and extract relevant information, thus creating a need for tools that can effectively extract and summarize the information. One such tool, is topic modelling, which is a method of extracting hidden themes or topics in a large collection of documents.

Information is stored in many forms, but of particular interest is the information stored as short text, which typically arises as posts on websites like *Facebook* and *Twitter* where people freely share their ideas, interests and opinions. With such a wealth in data and so many diverse users, such stores of short text could potentially provide useful information about public opinion and current trends, for instance. Unlike long text, like news and journal articles, one of the commonly known challenges of applying topic models on short text is the fact that it contains few words, which means that it may not contain sufficiently many meaningful words.

The Latent Dirichlet Allocation (LDA) model is one of the most popular topic models and it makes the generative assumption that a document belongs to many topics. Conversely, the Multinomial Mixture (MM) model, another topic model, assumes a document can belong to at most one topic, which we believe is an intuitively sensible assumption for short text. Based on this key difference, we posit that the MM model should perform better than the LDA.

To validate this hypothesis we compare the performance of the LDA and MM on two long text and two short text corpora, using coherence as our main performance measure. Our experiments reveal that the LDA model performs slightly better than the MM model on long text, whereas the MM performs better than the LDA model on short text.

ACKNOWLEDGEMENTS

I would like to extend my deepest gratitude to my supervisor, Dr A de Waal, for her unwavering support and guidance. I am sincerely grateful for her invaluable insight, the time and effort she dedicated to this work and, especially, for her contagious enthusiasm.

I would also like to thank my co-supervisors, Mr S Millard and Dr F Kanfer, for their wise counsel and constant support.

I am also grateful to the Department of Statistics at the University of Pretoria for presenting me with this research opportunity as well as the National Research Fund of South Africa (NRF) and the Centre for Artificial Intelligence Research (CAIR), an affiliate of the Council for Scientific and Industrial Research (CSIR), for the financial support required for the completion of this research.

I would like to thank my family and friends for always believing in me and supporting me in all my endeavours. A special thanks to my parents, Elsie and Upenyu Mazarura for their encouragement, love and prayers. I am also grateful to my siblings, Grace, Esther and Michael Mazarura for their constant enthusiasm and optimism. And, to my closest companion, Tinashe Mavindidze, my sincere gratitude for all the love, support and faith.

Above all, I would like to thank God, without whom none of this would have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CHAPTER ONE

INTRODUCTION

Topic modelling is a text mining technique that is used to uncover the underlying hidden topics or themes from a large archive of documents. In addition, it also enables one to cluster documents based on thematic similarity. Documents may be news articles, eBooks, webpages or tweets (posts from *Twitter*) that may need to be analysed. The power in unsupervised topic modelling lies in the fact that it is an unsupervised learning technique and can be used for the purposes of information retrieval and classification on large sets of data that would be impractical for humans to do manually.

1.1 MOTIVATION

Latent Dirichlet Allocation (LDA) is one of the most popular topic models and over the years it has proven to be very successful when applied to long text, such as news articles and academic abstracts (Mehrotra *et al.*, 2013). In recent times, there has arisen a surge of interest in the analysis of short text, such as user comments on websites as well as microblogs, which typically arise from social media services, such as *Twitter* [1] and *Facebook* [2]. This is due to the realisation that such

[1] `https://twitter.com`
[2] `https://www.facebook.com`

posts on websites could hold information that could be potentially useful for sentiment analysis (Lin and He, 2009), prediction purposes (Bollen *et al.*, 2011; Bermingham and Smeaton, 2011) and product marketing (Xiang and Zhou, 2014), for instance. However, unlike long text, the fact that short text is made up of few words implies that even fewer of those words will be useful in describing the hidden topic structure of the corpus. Consequently, this poses considerable problems when applying traditional topic models.

1.2 OBJECTIVES

The purpose of this work is to understand the performance of probabilistic topic models on short text such as microblogs and tweets. We compare two topic models - the Multinomial Mixture (MM) and Latent Dirichlet Allocation. We hypothesize that the simpler MM topic model will perform better on short text than its more popular counterpart, LDA. The latter topic model makes the generative assumption that each document is derived from multiple different topics with varying probabilities. On the other hand, the simpler MM topic model assumes that each document is associated with only a single topic, which we believe to be an intuitively sensible assumption for short text, such as a tweet or Facebook status.

Ultimately, we hope to answer the following question:

> *Given the assumption of each document belonging to at most one topic, can the hidden themes in a short text corpus be identified better by the Multinomial Mixture model than by its more sophisticated counterparts in a practical setting?*

1.3 CONTRIBUTIONS OF THIS WORK

To our knowledge, such a comparison of the MM and LDA models has not been done before. We hope that our hypothesis will be supported through our experimentation as this will provide researchers with a simple and viable tool for handling short text.

In the event that the results are not favourable, this text will provide a unified study of the MM model and provide a greater understanding of the challenges that entail handling short text. Furthermore, on seeing whatever shortcomings that the Multinomial Mixture model may possess, we will be in a position to propose possible solutions and modifications that we can investigate in future studies.

1.4 DISSERTATION STRUCTURE

The dissertation is structured as follows:

- Chapter 2 introduces topic models in more detail as well as presents some examples of the different topic models that exist. We also present a discussion on short text and its significance.
- Chapter 3 is an introductory chapter to some of the concepts that will be used. It presents the theory behind mixture models as well as the main estimation procedures that will be applied later on. The Expectation-Maximization (EM) algorithm and Gibbs sampler are both presented along with examples on simulated Gaussian data.
- Chapter 4 is the core of our study and presents a detailed study of the Multinomial Mixture model and its application in the context of topic models.
- Topic model evaluation is an area of much significance and challenge due to the fact that topic models are unsupervised learning techniques. There is no consensus as to which is the best evaluation method so, in Chapter 5 we discuss some of the evaluation methods that have been proposed.
- The experiments that were performed as well the results are then presented in Chapter 6. The experiments are designed in order to assess the model training convergence of each model as wall as measuring topic quality.
- Finally, the conclusions and discussion of future work are presented in Chapter 7.

CHAPTER TWO

TOPIC MODELLING

2.1 INTRODUCTION

As technology continues to rapidly advance, our capability to collect and store information digitally has also increased vastly. Information is stored in many forms, such as news articles, emails, scientific articles, webpages, eBooks and microblogs on social networks. The sizes of such collections of electronic information are continually growing at very high rates resulting in massive stores of data. Consequently, it becomes increasingly difficult to find and extract relevant information from such sources. This has fostered the need to develop efficient learning techniques that enable us to extract, summarise and understand the information contained within them (De Waal, 2010).

There are several learning techniques that have been proposed for the analysis of large document collections and they can usually be categorized as unsupervised or supervised. In the supervised learning context, documents are classified into predefined classes. Although this is useful in some situations, unfortunately there is often little, if any, prior knowledge about the information contained in the documents so such techniques are not always applicable. In such cases, unsupervised learning techniques must be used. These techniques allow the user to classify documents without the use of predefined categories or labels.

Clustering is a widely used unsupervised technique that, when applied to documents, groups similar themed documents together and separates them from those based on different topics (Huang *et al.*, 2009). This is achieved by representing each document as a high-dimensional vector of word frequencies (the *bag-of-words* representation) and applying standard vector-based clustering techniques such as k-means and agglomerative clustering. One of the weaknesses of this method is that, although documents are clustered into groups according to their *thematic* or *semantic* similarities, it is still not known what topic each cluster represents. In addition, clustering techniques make the assumption that each document can only belong to one topic (Newman *et al.*, 2006). However, in some situations, particularly with long text, such as news and journal articles, it is more realistic to assume that a document is about more than one topic in varying degrees.

Topic modelling is a special text clustering technique that possesses the advantage of addressing the former weakness of standard clustering methods. By definition, topic modelling is a text mining technique that enables one to uncover the underlying hidden topics or themes from a large archive of documents, in addition to clustering them based on thematic similarity. This topic identification is achieved by assuming that each document was formed through some generative process. A generative process can be thought of as the imaginary process by which documents are assumed to have been created (Blei *et al.*, 2003). There are different types of topic models and some even offer the user the option of assuming that documents can contain more than one topic.

2.2 VISUALIZING TOPIC MODELS

To illustrate the concept of topic models and provide a better understanding, we give an overview of a specific topic model, Latent Dirichlet Allocation (Blei *et al.*, 2003), by following an example from the article titled *Probabilistic topic models* by Blei (2012). Latent Dirichlet Allocation assumes that a document can contain multiple different topics in different proportions. A topic is regarded as a distribution over words, thus each topic is described by certain words, with some words having higher probabilities than others. Consider the following snippet from an article titled *Seeking Life's Bare (Genetic) Necessities* (Pennisi, 1996). The article is about determining the number of genes required for the survival of an organism.

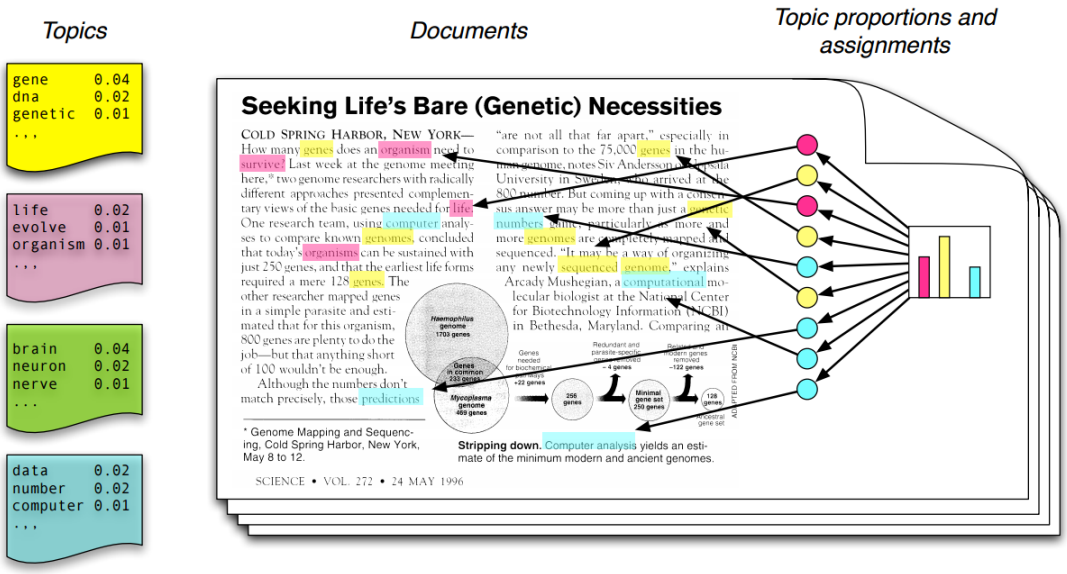


Figure 2.1: Illustration of LDA from Blei (2012)[1]

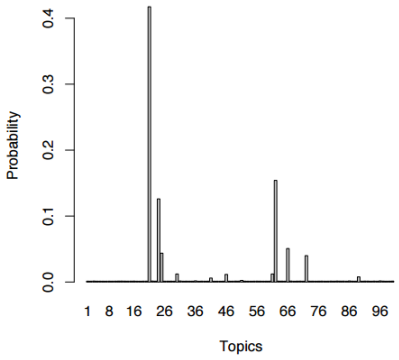
The far left side of Figure 2.1 shows example topics with the top three words with the highest probability of belonging to each topic. Looking at the top words in the yellow topic, *gene*, *dna* and *genetic*, it appears the topic is *genetics*. Similarly, the pink, green and blue topics appear to describe the topics *evolutionary biology*, *nervous system* and *data analysis*, respectively.

Assuming each word belongs to at least one topic, each word in the document in Figure 2.1 can be highlighted according to the colour of the topic to which it belongs. For instance, *genes* and *genomes* are highlighted in yellow, whereas words like *computer* and *predictions* are highlighted in blue. In this document, words were highlighted manually, but suppose each word, excluding non-informative words like *and*, *but* and *if*, was highlighted according to its topic, the document would be seen to contain different topics in various proportions. These proportions are represented by the histogram on the right of Figure 2.1, from which it can be seen that this document is about *genetics*, *evolutionary biology* and *data analysis*.

The main task of a topic model is to uncover the hidden topic structure given the observed documents. Figure 2.2 shows typical results from a topic modelling algorithm. The model was trained on 17 000 articles from *Science* magazine and then used to infer the topic structure of the

[1]Copyright ©2012 ACM, Inc. Included here by permission

example article from Figure 2.1 under the assumption that the training set contained 100 topics.



"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Figure 2.2: Example results from applying a topic model from Blei *et al.* (2003)[2]

The histogram shown on the left side of Figure 2.2 shows the inferred topic distribution for the example article. It can be seen that although there were 100 possible topics that the article could have contained, only some of the topics featured in the article and with varying proportions.

When a topic model is run, the outcome for each topic will be a set of words with varying probabilities. It then requires a person to look at the words in the set and decide on the name of the topic that the words could represent. Four of the most probable topics for the example article are shown on the right of Figure 2.2. The bold topic names at the top of each list are topic labels created by a person, given the list of words below. The lists represents the top 15 most probable words in each specific topic.

Although we have only looked at the LDA here, other probabilistic topic models can be imagined in a similar manner, given their generative processes and assumptions. As mentioned before, topic models do not assume any prior knowledge of what the corpus or documents are about. They are applied based on the premise that the inferred hidden structure of the corpus represents its thematic structure. Consequently, topic models provide a tool that can be useful for information retrieval, classification and corpus exploration on a scale so large that it would be unrealistic to do manually.

[2]Copyright ©2012 ACM, Inc. Included here by permission.

2.3 EXAMPLES OF TOPIC MODELS

This section presents an overview of some of the different topic models that exist, as well as gives some of the details of their individual generative processes. In order to easily see some of the differences between the different models their respective graphical models are also given. Figure 2.3 shows an example of a directed graph with its interpretation.

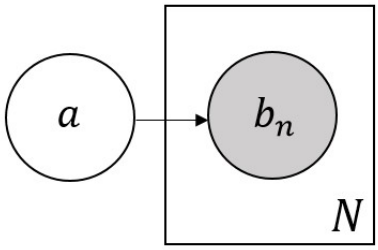


Figure 2.3: An example of a directed graph

- **circles:** The circles, called nodes, denote random variables, such as, variables a and b_n. A shaded node indicates that the random variable that is associated with the node is observed, for example b_n, whereas an unshaded node indicates a variable that is not observed, like a.
- **arrow:** The arrow between nodes is called an edge. They represent a possible dependency between the two variables. From the graph it can be seen that, b_n depends on a.
- **rectangle:** The rectangle is called a plate and denote a replicated structure. For instance, the N-plate indicates that b_n is generated N times.

Many text clustering approaches exist, each with its own set of assumptions, but we will focus on *probabilistic* topic models. Such methods assign a probability of a document belonging to a specific topic by assuming documents are formed probabilistically and can be described by either its assumed generative process or graphical model. These techniques include Probabilistic Latent Semantic Analysis (Hofmann, 1999*b*), the Gamma-Poisson model (Canny, 2004), Latent Dirichlet Allocation (Blei *et al.*, 2003) and the Multinomial Mixture model (Nigam *et al.*, 2000).

2.3.1 PROBABILISTIC LATENT SEMANTIC ANALYSIS

LATENT SEMANTIC ANALYSIS (LSA)

Assuming documents are represented as high dimensional vectors of word frequencies, the entire collection of documents (corpus), can be regarded as a $document \times word$ matrix. Latent Semantic Analysis (Deerwester *et al.*, 1990) performs a dimension reduction on the (document) frequency vector by projecting the vector onto a lower dimensional vector (latent) space that captures most of the variance of the corpus (Hofmann, 1999*b*; Blei, 2004). The mapping is restricted to be linear and is determined by applying a Singular Value Decomposition (SVD) on the $document \times word$ matrix. This dimension reduction reduces the sparcity of the vectors and should map terms with similar meanings onto vectors with the same direction in the latent space (Hofmann, 2001). As a result, it is possible to find meaningful relationships between thematically similar documents even if they do not contain any of the same words (Hofmann, 1999*a*). Despite this benefit and the fact that it has proven to be successful in many fields, the disadvantage of LSA is that it lacks a solid statistical foundation and therefore, lacks a clear probabilistic interpretation (Hofmann, 1999*a*). In view of this weakness, Hofmann (1999*b*) devised a new model based on LSA which has a sound statistical basis with a clearly defined generative model for the data, Probabilistic Latent Semantic Analysis. Having a statistically sound basis implies that standard statistical model selection and fitting procedures can be applied (Hofmann, 2001).

PROBABILISTIC LATENT SEMANTIC ANALYSIS

Probabilistic Latent Semantic Analysis (pLSA) still makes use of SVD for dimension reduction of the $document \times word$ matrix. It assumes the bag-of-words representation of documents. Moreover, according to Hofmann (1999*b*), pLSA also assumes the following generative process:

1. Assuming the d-th document is represented by a vector of word counts, $\vec{d}$, select a document with probability $P(d)$, $(d = 1, 2, ..., D)$.

2. For each word, w_n, $(n = 1, 2, ..., N)$ in the d-th document:

   a) Pick a latent class (topic) $z_{d,n}$ from a multinomial with probability $P(z|d)$.

   b) Generate a word $w_{d,n}$ from a multinomial with probability $P(w|z)$.

The generative process can be represented graphically as in Figure 2.4.

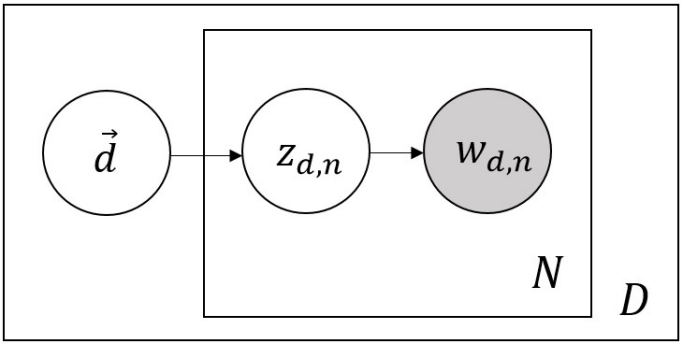


Figure 2.4: pLSA graphical model

The model is then expressed as

$$P(d, w) = P(d) \sum_{z \in \mathcal{Z}} P(w|z)P(z|d).$$

The parameters are chosen in such a way as to maximize the log-likelihood of the model, but due to the presence of latent variables the estimation is performed using the Exponential-Maximization (EM) algorithm. The reader is referred to Hofmann (1999*a*) for an in-depth presentation of this modelling procedure.

2.3.2 GAMMA-POISSON MODEL

The Gamma-Poisson (GaP) model is a probabilistic model that was proposed by Canny (2004), to simultaneously address the need for accurate search and retrieval of information from a corpus, topic identification and clustering of documents according to thematic similarity Canny (2004). As was first described in by Canny (2004), the model is summarized as follows:

Assuming the $document \times word$ matrix, $F = (F_{ij})$, contains the number of occurrences of word i in document j, the aim of the model is to find an approximate factorization of F. Letting the matrix Y denote the expected value of F, the factorization is approximated by

$$Y = \Lambda X.$$

Each element, X_{ij}, in the $topic \times document$ matrix X is a non-negative random variable indi-

cating the contribution of topic i in document j. The matrix Λ denotes the $word \times topic$ matrix, where each element, λ_{ij}, denotes the probability of word i in topic j. The matrix factorisation is visualized as shown in Figure 2.5.

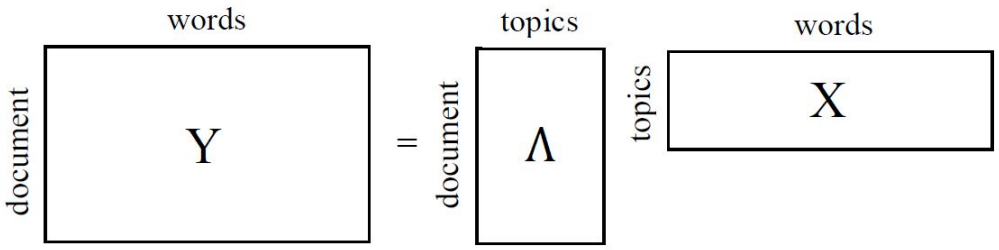


Figure 2.5: Matrix Factorisation for GaP

It follows from the definition of X, that its d-th column, $\vec{x}_d = (x_{d1}, x_{d2}, ..., x_{dT})'$ $(d = 1, 2, ..., D)$, contains the topic weights of the t-th topic in document d, x_{dt} $(t = 1, 2, ..., T)$. Similarly, if $\vec{y}_d = (y_{d1}, y_{d2}, ..., y_{dV})'$ denotes a column of Y then it is a vector containing the expected frequency of each of the V words in document d. The name of the model, Gamma-Poisson, is derived from the fact that the first random variable in the generative process, x_{dt}, follows a Gamma distribution, whilst the last random variable, y_{dw} $(w = 1, 2, ..., w)$, follows a Poisson distribution.

The generative process is summarized as follows:

1. For each document, $\vec{x}_d$ $(d = 1, 2, ..., D)$, independently draw $x_{dt} \sim \text{Gamma}(a_t, b_t)$ where a_t and b_t $(t = 1, 2, ..., T)$ denote the shape and scale parameters of the Gamma distribution, respectively.

2. For each word w in document d, draw counts $y_{dw} \sim \text{Poisson}(\vec{\lambda}_w \vec{x}_d)$ where $\vec{\lambda}_w$ denotes the w-th row of Λ $(w = 1, 2, ..., V)$.

The parameters of the model are then estimated using the EM algorithm. The reader is referred to Canny (2004) for further details regarding this model.

Graphically, the generative process can be represented as in Figure 2.6.

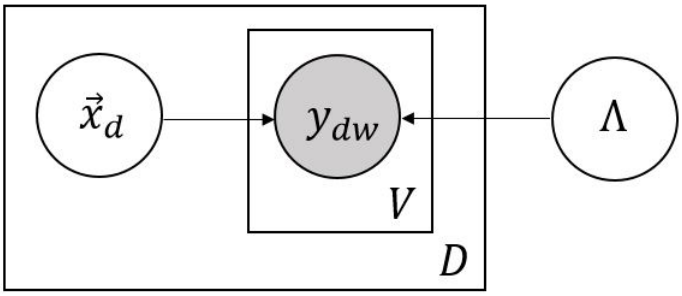


Figure 2.6: GaP graphical model

2.3.3 LATENT DIRICHLET ALLOCATION

The Latent Dirichlet Allocation (Blei *et al.*, 2003) model is one of the most popular probabilistic topic models. LDA assumes that the corpus contains a certain number of topics, T, and makes the assumption that each document is formed through a generative process as shown in Blei *et al.* (2003):

1. Randomly choose T topic distributions, $\beta_t \sim Dirichlet(\vec{\lambda}_\beta)$[3]. Letting $\beta = (\vec{\beta}_1, \vec{\beta}_2, ..., \vec{\beta}_T)$ denote the $topic \times word$ matrix, each elements, β_{ij} represents the probability of the i-th word belonging to the j-th topic.

2. For each document, $\vec{d} = (w_1, w_2, ..., w_N)$, in the corpus:

   a) Randomly choose a distribution over topics, $\vec{\theta}_d \sim Dirichlet(\vec{\lambda}_\alpha)$.

   b) For each of the N words, w_n, in document d:

      i. Randomly choose a topic $z_n \sim Multinomial(\vec{\theta}_d)$.

      ii. Randomly choose a word $w_n \sim Multinomial(\vec{\beta}_{z_n})$.

[3]The K-dimensional Dirichlet distribution has the following probability density function,

$$p(\theta|\vec{\alpha}) = \int \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1},$$

where $\Gamma(x)$ denotes the Gamma function, $\vec{\theta}$ and $\vec{\alpha}$ are both K-dimensional vectors whose elements have the following properties: $\theta \geq 0$, $\sum_{k=1}^{k} \theta_i = 1$ and $\alpha_i > 0$. The Dirichlet distribution is a member of the exponential family of distributions, has finite dimensional sufficient statistics and is conjugate to the multinomial distribution (Blei *et al.*, 2003).

Figure 2.7 represents this graphically.

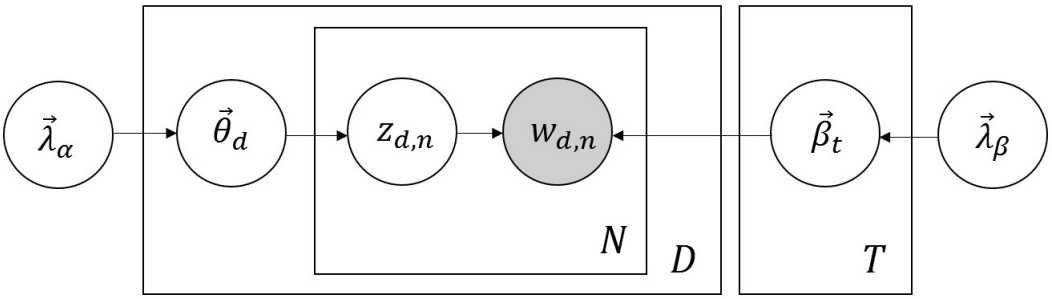


Figure 2.7: The LDA graphical model

In order to use the LDA, the computation of the posterior distribution of the hidden variables given a document is the key inferential problem:

$$p(\vec{\theta}, \beta, \vec{z}|\vec{d}, \vec{\lambda}_\alpha, \vec{\lambda}_\beta) = \frac{p(\vec{\theta}, \beta, \vec{z}, \vec{d}|\vec{\lambda}_\alpha, \vec{\lambda}_\beta)}{p(\vec{d}|\vec{\lambda}_\alpha, \vec{\lambda}_\beta)}$$

where $\vec{z} = \{z_1, z_2, ..., z_T\}$ and $\beta = (\vec{\beta}_1, \vec{\beta}_2, ..., \vec{\beta}_T)$. Unfortunately, this is not tractable as the normalization constant in the denominator can not be computed exactly.

The marginal distribution of a document, assuming the LDA model, is:

$$p(\vec{d}|\vec{\lambda}_\alpha, \beta) = \frac{\Gamma(\sum_k \lambda_\alpha)}{\prod_k \Gamma(\lambda_\alpha)} \int \left(\prod_{k=1}^{T} \theta_k^{\lambda_\alpha - 1} \right) \left(\prod_{n=1}^{N} \sum_{i=1}^{T} \prod_{j=1}^{V} (\theta_i \beta_{ij})^{w_n^j} \right) \mathrm{d}\theta. \tag{2.1}$$

Fortunately, estimation methods such as variational Bayes and the Gibbs sampler are applicable approximation algorithms. The reader is referred to Blei *et al.* (2003) for a detailed description of the model.

2.3.4 MULTINOMIAL MIXTURE MODEL

Under this model, topics are assumed to follow a multinomial distribution over words, consequently the corpus is modelled as a mixture of multinomials. Looking at the amount of available literature, the Multinomial Mixture model appears to be a less popular probabilistic generative model, probably due to the fact that that it makes the somewhat unrealistic assumption that each document in a corpus can only belong to a single topic. We believe that this perceived weakness

could be a strength in some situations. This model is the main focus of this work and will be discussed in detail in Chapter 4.

2.4 SHORT TEXT

Traditionally, when we refer to long text we are typically referring to texts such as scientific journal articles, books and news articles. The growing popularity of the internet has resulted in the emergence of online publishing, instant messaging, e-commerce and social media (Jun *et al.*, 2013). With the increasing accessibility of the internet over the past few years, a plethora of textual data stores has been created and is continually growing. Of particular and notable interest, is the information available as short text. As the name describes, in contrast to long text, short text is short and consequently, is more sparse than long text. Examples include posts on *Twitter* (called tweets) which are restricted to a mere 140 characters (Hong and Davison, 2010) and user comments on Hellopeter.com[4]. One of the well-known challenges of handling short text is the fact that it contains few words, which means that there may not be sufficiently enough words to aid the process of topic assignment.

Over the years, there has been a significant increase in the amount of short text available, as social networks like *LinkedIn*[5], *Facebook* and *Twitter*, have risen in popularity. Such data stores often contain information that is relevant and potentially valuable, especially at the time when the posts are created as these websites are growingly used for communicating breaking news and eyewitness accounts (Hong and Davison, 2010), as well as for sharing ideas (Khartabil, 2013). Consequently, since the realisation that possibly useful information can be extracted through the analysis of such data stores, the need for efficient techniques for information extraction has become increasingly relevant, especially for short text.

The application of information retrieval tools on tweets has become a subject of much interest in recent times as they can provide an indication of current trends, public interests and public reaction to breaking news (Khartabil, 2013). Such information can then be used for sentiment analysis and prediction purposes, amongst other uses. For instance, a person in business may use sentiment analysis for quality control purposes or to aid in the development of products that

[4] `http://hellopeter.com`
[5] https://www.linkedin.com

address consumer needs. Likewise, Bermingham and Smeaton (2011) studied the relation between tweets and the results of the Irish General Election and concluded that the tweets had predictive qualities. In another study, Bollen *et al.* (2011) investigated the relationship between the public mood and some economic indicators. They analysed *Twitter* feeds from two mood tracking tools and found that the use of some of the information helped improve the accuracy of the predictions on the daily fluctuations in the closing values of the Dow Jones Industrial Average. Lastly, Gerber (2014) found that, compared to a standard approach, by incorporating *Twitter* data into a crime prediction model the prediction performance of the model was improved for some types of crime in the United States.

As previously mentioned, due to its sparcity and limited information content, short text presents challenges when applying typical topic models. However, much research is being conducted to find methods and techniques that can be implemented to effectively handle short text. One such method is pooling (Mehrotra *et al.*, 2013; Hong and Davison, 2010). This involves merging multiple short texts to form a single longer document and then applying the topic modelling procedure as normal. In the case of tweets, they can be pooled according to user or according to hash tags (user defined tags staring with "#" that are used to identify topics or events). Upon comparing the different pooling schemes, Mehrotra *et al.* (2013) found that pooling according to user was better than not pooling at all and pooling according to hash tag gave even better results than pooling according to user.

2.5 DATA VISUALISATION

Apart from topic models, there exist other approaches to text analysis. Data visualization is a way of describing data using visual aids such as graphs and charts with the aim of helping viewers quickly analyse and understand the data (Khartabil, 2013). In the context of text analysis, *word clouds* are an example of one such visualization tool. A document or any such body of text is represented as a cluster of words, with possibly various fonts, colours and sizes.

Figure 2.8 is an example of a word cloud that was formed based on President Obama's 2011 state of Union speech.

Figure 2.8: Word cloud formed from President Barack Obama's 2011 state of Union speech (Khartabil, 2013)

The more frequent a term in the body of text appears, the larger and bolder the word will appear in the word cloud. Unfortunately, the frequency of a word does not necessarily reflect the importance of the word. Other weaknesses of word clouds include that it is difficult to compare documents and that it is unable to capture how a topic varies over time. Fortunately, solutions to these problems have been proposed. The reader is referred to Castella and Sutton (2014) and Cui *et al.* (2010) for the details of *word storms* and *dynamic word clouds*, respectively.

Topics in temporally related documents can also be analyzed through the use of TopicRivers (Havre *et al.*, 2000). The name comes from the fact that the representation looks like a river. Figure 2.9 is an example of a TopicRiver from (Havre *et al.*, 2000). It is based on an associated press (AP) newswire corpus of 100 000 documents from July to August 1990. It flows from left to right through time with varying width to represent the variation in the thematic strength of the documents over time. Each colour represents a different topic whilst the width represents the strength of the topic. A wide width indicates heavy use of a topic whilst changes in topics are indicated by changes in colour (Havre *et al.*, 2000). For a good and comprehensive summary of other such data visualizations the reader is referred to Khartabil (2013).

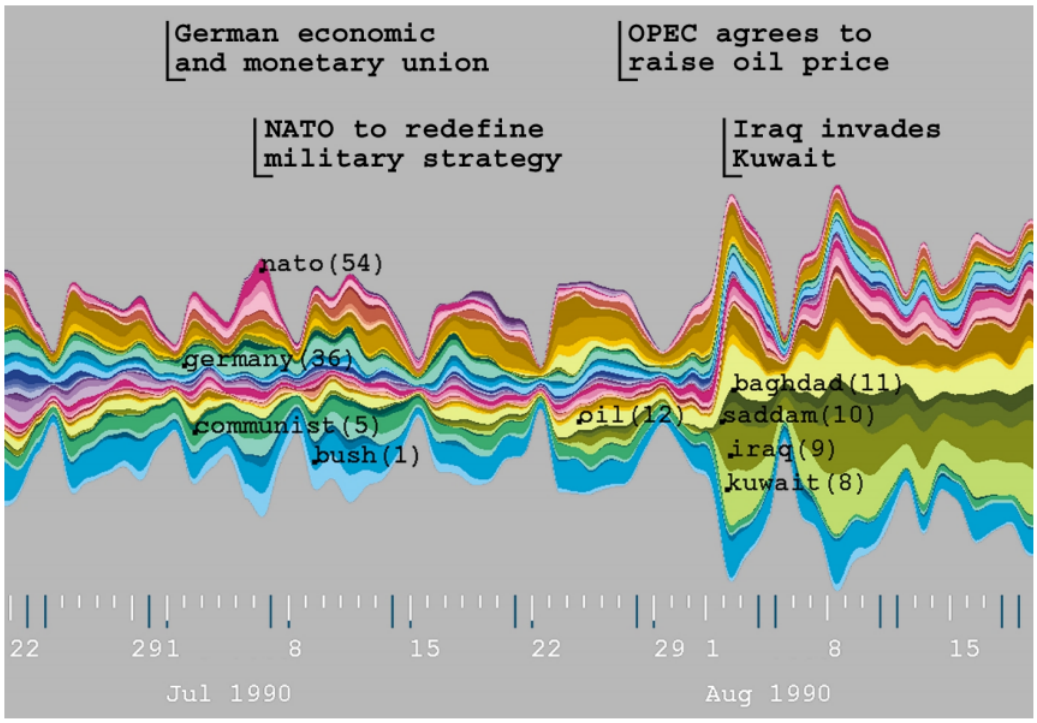


Figure 2.9: ThemeRiver formed from AP corpus from July to August 1990 from Havre *et al.* (2000)[7]

2.6 PROBLEM STATEMENT/HYPOTHESIS

This body of work aims to find a viable manner of handling short text and overcoming the challenges faced by traditional topic models, specifically Latent Dirichlet Allocation. The LDA assumes that a document can belong to many topics with some distribution of topics. Intuitively, it seems short text, such as a tweet, would more likely be from a single topic rather than from many topics. Although the Multinomial Mixture model would be expected to perform poorly on long text, we believe that it would be more advantageous to model a document according to the Multinomial Mixture model which makes the assumption that a document belongs to at most one topic. The objective of this work is to investigate this hypothesis and hopefully find supporting evidence to promote the use of the Multinomial Mixture model for short text.

[7]Copyright ©2000 IEEE. Included here with permission.

CHAPTER THREE

PRELIMINARIES: PARAMETER ESTIMATION

3.1 INTRODUCTION

Although the main focus of this body of work is to investigate topic modelling in the context of the Multinomial Mixture model, it is necessary to first introduce some of the mathematical concepts that will need to be used in developing the theory. In this section, we explore mixture models and how they are formed. This will then be followed by a detailed discussion of how the respective parameters are estimated for specific cases as well as a few simple examples based on Gaussian data.

3.2 THE K-COMPONENT MIXTURE MODEL

A mixture model is a linear combination of models. Such models are most applicable where one intends to model a population which is believed to have originated from a mixture of different homogeneous subpopulations (Chen *et al.*, 2004). It is assumed that each subpopulation follows a different model and that the data arises from these models in different proportions (Lindsay, 1995).

Let Y_i denote a random variable with probability density function (pdf) $f(y_i)$ where θ is its respective set of parameters and y_i, its realization. In addition, let $Y = \{Y_1, Y_2, ..., Y_N\}$ denote a

random sample of size n. In the context of mixture modelling, the probability density function of Y, assuming there are $K \in \mathbb{R}$ models is

$$g(y_i) = \sum_{k=1}^{K} \pi_k f_{\theta_k}(y_i),$$

where K is fixed, $\sum_{k=1}^{K} \pi_k = 1$ and $0 \leq \pi_k \leq 1$. The π_k are referred to as the mixing proportions or weights and θ_k denotes the parameter(s) associated with the respective probability density function. Thus, $\theta = \{\pi_1, ..., \pi_K, \theta_1, ..., \theta_K\}$ denotes the vector of parameters. (Hastie *et al.*, 2009)

In order to fit the model fully, the parameters must be estimated. Maximum likelihood estimation, would require one to maximize the log-likelihood. Typically, it is unknown from which model each observation in the dataset was originally derived. Since no variable indicating the origins of each observation is observed the data set based on the y_i's is referred to as the incomplete dataset. The log-likelihood function based on the *incomplete* data set is

$$\log L(\theta|\mathbf{y}) = \log \prod_{i=1}^{N} g(y_i)$$

where $\theta = \{\pi_1, ..., \pi_K, \theta_1, ..., \theta_K\}$. Unfortunately, due to the summation inside the logarithm it is not possible to maximize directly, thus it becomes necessary to employ other methods, such as the EM algorithm (Hastie *et al.*, 2009). In order to apply the EM algorithm, a hidden variable is introduced. This variable is assumed to be unobservable and is referred to as the *latent* (hidden) variable, denoted

$$Z_{ik} = \begin{cases} 1 & \text{if } y_i \text{ is from model } k \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, 2, ..., N$ and $k = 1, 2, ..., K$ with $P(Z_{ik} = 1) = \pi_k$. The dataset, $\{Y, Z\}$, which includes the latent variable is referred to as the *complete* data set.

Since the pdf of y_i given z_i is

$$p(y_i|z_i) = \prod_{k=1}^{K} f_{\theta_k}(y_i)^{z_{ik}}$$

and the pdf of z_i is

$$p(z_i) = \prod_{k=1}^{K} \pi_k^{z_{ik}}$$

it follows that the pdf of the complete data set can be written as

$$p(y_i, z_i) = \prod_{k=1}^{K} (\pi_k f_{\theta_k}(y_i))^{z_{ik}}$$

after applying Bayes theorem.

Therefore, the log-likelihood based on the complete data set is as follows:

$$\begin{aligned} \log L(\theta|y, z) &= \log \prod_{i=1}^{N} p(y_i, z_i|\theta) \\ &= \log \prod_{i=1}^{N} \prod_{k=1}^{K} (\pi_k f_{\theta_k}(y_i))^{z_{ik}} \\ &= \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \log(\pi_k f_{\theta_k}(y_i)) \end{aligned}$$

where $f_{\theta_k}(y_i)$ can be any probability density (or mass) distribution function. In order to formulate the K-component Gaussian (normal) mixture model $f_{\theta_k}(y_i)$ is replaced by the probability density function of the normal distribution. Similarly, as will become relevant later, $f_{\theta_k}(y_i)$ is replaced by the probability mass function of the multinomial distribution to form a K-component Multinomial Mixture model.

In the following section we discuss the EM algorithm. We look at its application to the Gaussian mixture model which is well documented in the literature (see, for example, Bishop *et al.*, 2006). We will then consider the Gibbs sampler, as another competing method that can be used to derive the required estimates.

3.3 THE EM ALGORITHM

The EM algorithm is useful when the maximization of a likelihood function is difficult, but can be simplified by the introduction of latent data. This is known as *data augmentation*. In the mixture model context the latent data are the model memberships, Z_{ik}.

Suppose Y is the observed data having log-likelihood, $\log L(\theta|Y)$, where θ denotes the pa-

rameters. If Z denotes the latent variable, then the complete data set is $\{Y, Z\}$ and the complete log-likelihood is $\log L(\theta|Y, Z)$. The EM algorithm iteratively estimates the maximum likelihood estimates of the parameters in two main stages:

1. The Expectation-step (E-step): The j-th parameter estimates, $\hat{\theta}^{(j)}$, is used to find the expected value of the posterior distribution of the latent variables, $p(Z|Y, \hat{\theta}^{(j)})$. This expected value is also referred to as the *responsibility*. The responsibilities are then used to calculate the expected value of the complete log-likelihood.

2. The Maximization-step (M-step): The new parameter estimates $\hat{\theta}^{(j+1)}$ are then calculated by maximizing the expectation of the complete likelihood.

According to Hastie *et al.* (2009) the EM algorithm can be summarized as in Algorithm 1.

Algorithm 1: The EM Algorithm

1 Initialise $\hat{\theta}^{(0)}$.
2 *E-step:* at the j-th step determine

$$Q(\theta', \hat{\theta}^{(j)}) = E(\log L(\theta|Y, Z)|Z, \hat{\theta}^{(j)})$$

where θ' denotes the dummy argument.
3 *M-step:* determine the maximizer of $Q(\theta', \hat{\theta}^{(j)})$ over θ'. This maximizer is the new estimate $\hat{\theta}^{(j+1)}$.
4 Repeat steps 2 and 3 until the iteration converges.

3.3.1 APPLICATION OF THE EM ALGORITHM TO THE K-COMPONENT GAUSSIAN MIXTURE MODEL

Firstly, the means, $\hat{\mu}_k$, are initialised to random observations, the variances, $\hat{\sigma}_k^k$, to $\sum_i^N \frac{(y_i - \bar{y})^2}{N}$ and the responsibilities, denoted γ_{ik}, to $\frac{1}{N}$.

Then, in the E-step, the responsibilities are computed:

$$\begin{aligned} \gamma_{ik} &= E(z_{ik}|y_i, \boldsymbol{\theta}) \\ &= P(z_{ik} = 1|y_i, \boldsymbol{\theta}) \\ &= \frac{\pi_k f_{\theta_k}(y_i)}{\sum_{l=1}^K \pi_l f_{\theta_l}(y_i)}. \end{aligned} \tag{3.1}$$

In the M-step, the complete likelihood must be maximized with respect to $\theta = \{\pi_1, ..., \pi_K, \theta_1, ..., \theta_K\}$.

Optimization with respect to π_k for $k = 1, 2, ..., K$: replacing the latent variable with the responsibilities:

$$\frac{\partial}{\partial \pi_k} \log L(\theta|\mathbf{y}, \mathbf{z})) = \frac{\partial}{\partial \pi_k} \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \log(\pi_k f(y_i|\theta_k)) = 0$$

$$\Rightarrow \pi_k = \sum_{i=1}^{N} \frac{\gamma_{ik}}{N}, k = 1, 2, ..., K. \tag{3.2}$$

Optimization with respect to (μ_k, σ_k^2) for $k = 1, 2, ..., K$: this is done in a similar manner to the previous optimization. After substituting the hidden variable in the complete log-likelihood with the responsibilities, the partial derivatives with respect to each parameter are computed and set to zero. It follows that

$$\hat{\mu}_k = \frac{\sum_{i=1}^{N} \gamma_{ik} y_i}{\sum_{i=1}^{N} \gamma_{ik}} \tag{3.3}$$

and

$$\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{N} \gamma_{ik} (y_i - \hat{\mu}_k)^2}{\sum_{i=1}^{N} \gamma_{ik}}. \tag{3.4}$$

The EM algorithm for the K-component Gaussian mixture model is shown in Algorithm 2.

Algorithm 2: The EM Algorithm for the K-component Gaussian mixture

1. Initialise $\hat{\mu}_k, \hat{\sigma}_k^2, \hat{\pi}_k$.
2. *E-step:* determine the responsibilities γ_{ik} for $i = 1, 2, ..., N$ as in (3.1)
3. *M-step:* compute the parameters $\hat{\pi}_k$, $\hat{\mu}_k$ and $\hat{\sigma}_k^2$ where $k = 1, 2, ..., K$ as in (3.2), (3.3) and (3.4) respectively.
4. Repeat steps 2 and 3 until the iteration converges.

EXAMPLE 1: APPLICATION OF THE EM ALGORITHM TO THE 3-COMPONENT GAUSSIAN MIXTURE MODEL

In this section we consider an example of how the EM algorithm is applied to a Gaussian mixture model with $K = 3$.

A dataset with 1 000 observation is generated from the following distributions, $Y_1 \sim N(3,1), Y_2 \sim N(-2,0.2)$ and $Y_3 \sim N(1.5,1)$, such that the proportion of observations from Y_1, Y_2 and Y_3 is $\pi_1 = 0.2, \pi_2 = 0.5$ and $\pi_3 = 0.3$ respectively. The SAS/IML®[1] code used to generate the this dataset is shown in Appendix A1.

The pdf of the mixture is therefore modelled as

$$g(y_i) = \pi_1 f_{\theta_1}(y_i) + \pi_2 f_{\theta_2}(y_i) + \pi_3 f_{\theta_3}(y_i)$$

where f_{θ_k} denotes the pdf of Y_k with parameters $\theta_k = (\mu_k, \sigma_k^2)$, $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{3} \pi_k = 1$, $k = 1, 2, 3$.

Assuming that the parameters $\theta = \{\pi_1, \pi_2, \pi_3, \mu_1, \mu_2, \mu_3, \sigma_1^2, \sigma_2^2, \sigma_3^2\}$ are unknown, their estimation is done using the EM algorithm as explained in Algorithm 2. This procedure is implemented using SAS/IML software and the code is shown in Appendix A2.

The initial values used for the mean were randomly selected observations from the dataset, $\hat{\mu}_1 = y_{185} = 1.8130894$, $\hat{\mu}_2 = y_{971} = -1.802821$ and $\hat{\mu}_1 = y_{400} = 0.7291342$. The initial values for the variances were set to $\hat{\sigma}_k^2 = \sum_{i=1}^{N}(y_i - \bar{y})^2/N = 5.0776604$ and the mixture proportions to $\hat{\pi}_k = 1/3$ for $k = 1, 2, 3$. The estimated values are updated iteratively by applying the E-step and M-step repeatedly until the absolute value of the difference between consecutive estimates of π_1 was less than 10^{-20}. The estimation procedure was completed after approximately 10 000 iterations. The estimated parameters were as shown in Table 3.1.

[1]Copyright, SAS Institute Inc. SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc., Cary, NC, USA.

Table 3.1: Table showing the estimated parameters from the EM algorithm

k	$\hat{\mu}_k$	$\hat{\sigma}_k^2$	π_k
1	3.3464100	0.7675018	0.1494513
2	-1.984433	0.2359508	0.5049559
3	1.5999679	0.8669275	0.3455928

From Table 3.1 it can be seen that the estimated values are close to the true values.

3.4 THE GIBBS SAMPLER

As was seen in the previous section, the EM algorithm maximizes over the conditional distributions. The Gibbs sampler (Geman and Geman, 1984), on the other hand, *samples* from these distributions, which is the main difference between these two estimation techniques (Hastie *et al.*, 2009). As described by Casella and George (1992), the Gibbs sampler is a technique that evades the need to calculate a density function by generating random variables from a (marginal) distribution indirectly. Ultimately, a sample $X_1, X_2, ..., X_m \sim f(x)$ is generated from which characteristics of $f(x)$, such as the mean or variance, can be calculated to the required degree of accuracy. As a result, in order to obtain the characteristics of $f(x)$ it is not necessary to calculate or approximate the distribution of X directly. In addition, any estimates derived from this process, although derived from simulations, are actually the population estimates (Casella and George, 1992).

Suppose it is required to draw a sample from the joint distribution of random variables $U_1, U_2, ..., U_K$ for which the marginal distributions $P(U_j|U_1, U_2, ..., U_{j-1}, U_{j+1}, ..., U_K)$ $(j = 1, 2, ..., K)$ can easily be simulated. Assuming some initial values, the Gibbs sampling procedure alternatively simulates an observation from each conditional distribution. As each distribution is sampled, the distribution of the variable is conditioned on all the most recent available observations (excluding itself). This process is repeated until the process stabilizes and the result is a sample from the required joint distribution (Hastie *et al.*, 2009).

According to Hastie *et al.* (2009) and Bishop *et al.* (2006) the algorithm can be summarized as follows:

Algorithm 3: The Gibbs sampler

1 Initialise $U_k^{(0)}$, $k = 1, 2, ..., K$.
2 Repeat for $t = 1, 2, ...$:

- Sample $U_1^{(t+1)} \sim P(U_1|U_2^t, U_3^t, ..., U_K^t)$
- Sample $U_2^{(t+1)} \sim P(U_2|U_1^{t+1}, U_3^t, ..., U_K^t)$

$\vdots$

- Sample $U_j^{(t+1)} \sim P(U_j|U_1^{t+1}, U_2^{t+1}, ..., U_{j-1}^{t+1}, U_{j+1}^t, ..., U_K^t)$

$\vdots$

- Sample $U_K^{(t+1)} \sim P(U_K|U_2^{t+1}, U_3^{t+1}, ..., U_{K-1}^{t+1})$

Repeat this step until the joint distribution of $(U_2^t, U_3^t, ..., U_K^t)$ remains unchanged.

3.4.1 APPLICATION OF THE GIBBS SAMPLER TO THE K-COMPONENT GAUSSIAN MIXTURE MODEL

We will follow the procedure as laid out in Hastie *et al.* (2009), and, for simplicity, we will fix the variances, $\{\sigma_k^2\}_{k=1}^K$, and mixing weights, $\{\pi_k\}_{k=1}^K$, to their maximum likelihood estimates. Consequently, the only parameters that will need to be estimated are the means $\{\mu_k\}_{k=1}^K$.

Unlike with the EM algorithm, the Gibbs sampler does not compute the maximum likelihood responsibilities as in the E-step, instead it samples from $p(Z|\theta, Y)$ where Z denotes the latent variable (label), Y denotes the data and $\theta = \{\mu_1, \mu_2, ..., \mu_K\}$. Furthermore, rather than finding the values of the parameters that maximize the posterior $p(\mu_1, \mu_2, ..., \mu_K, Z|Y)$ as in the M-step, it samples from the conditional distribution $p(\mu_1, \mu_2, ..., \mu_K|Z, Y)$.

Firstly, the means, $\theta^{(0)} = \{\mu_1^{(0)}, \mu_2^{(0)}, ..., \mu_K^{(0)}\}$, are initialized to zero. When running the Gibbs sampler, it takes some iterations for the sampler to stabilize so it is necessary to allow for an initial burn in period. This means that some of the initial samples must be discarded so that the effect of the initial values is eliminated. The class assignment label, $z_i = (z_{i1}, z_{i2}, ..., z_{iK})$, for each observation is sampled from a multinomial distribution with parameter $(\gamma_{i1}(\theta_k), \gamma_{i2}(\theta_k), ..., \gamma_{iK}(\theta_k))$ such that

$$\gamma_{ik}(\theta_k) = \frac{\pi_k f_{\theta_k}(y_i)}{\sum_{l=1}^{K} \pi_l f_{\theta_l}(y_i)} \tag{3.5}$$

where $f_{\theta_k}(y_i)$ is the probability distribution function of the normal distribution with parameter $\theta_k = (\mu_k, \sigma_k^2)$. The next step is to sample μ_k from a normal distribution with parameters $(\hat{\mu}_k, \hat{\sigma}_k^2)$ where $\hat{\sigma}_k^2 = 1$ and

$$\hat{\mu}_k = \frac{\sum_{i=1}^{N} \gamma_{ik} y_i}{\sum_{i=1}^{N} \gamma_{ik}}. \tag{3.6}$$

for $k = 1, 2, ..., K$.

The algorithm for the Gibbs sampler applied to a K-component Gaussian mixture model is shown in Algorithm 4.

Algorithm 4: The Gibbs sampler for the K-component Gaussian mixture

1 Initialise $\theta^{(0)} = (\mu_k^0)$ for $k = 1, 2, ..., K$.
2 Repeat for $t = 1, 2, ...$:

a) For $1 = 1, 2, ..., N$ generate $z_i^{(t)} \sim Multinomial(\gamma_{i1}(\theta^{(t)}), \gamma_{i2}(\theta^{(t)}), ..., \gamma_{iK}(\theta^{(t)}))$ where $\gamma_{ik}(\theta)$ is as defined in equation (3.5).

b) Compute $\hat{\mu}_k$ as in equation (3.6) and sample $\mu_k^{(t)} \sim N(\hat{\mu}_k, \hat{\sigma}_k^2)$.

Continue step 2 until the joint distribution of $(\mu_1, \mu_2, ..., \mu_K, Z)$ remains constant.

For a detailed derivation of the Gibbs sampler assuming all the parameters are unknown, the reader is referred to Yu (2009). In the case where all the parameters are unknown, Kamper (2013) shows that it is actually possible to simplify the Gibbs sampler by integrating out the model parameters μ_k, σ_k^2 and π_k from the posterior distribution and consequently, only using the Gibbs sampler to sample the labels, z_i. The other parameters can then be estimated given these assignments. This simplified model is called the *collapsed* Gibbs sampler or *Rao-Blackwellized* Gibbs sampler. The unsimplified model is therefore referred to as the *uncollapsed* Gibbs sampler.

EXAMPLE 2: GIBBS SAMPLER ON 3-COMPONENT GAUSSIAN MIXTURE MODEL

For this example, the Gibbs sampler, as described in Algorithm 4 is applied to the same dataset used in Example 1. The variances are set to all equal 1 whilst the mixing proportions are set to their maximum likelihood estimates as found in the EM algorithm, $\hat{\pi}_1 = 0.1494513$, $\hat{\pi}_2 = 0.5049559$ and $\hat{\pi}_3 = 0.3455928$. The initial values are $\mu_1^{(0)} = \mu_2^{(0)} = \mu_3^{(0)} = 0$ and the sampler is run for 10 000 iterations. The estimated values, $\hat{\mu}_k$, against the true values, μ_k, are shown in Table 3.2.

Table 3.2: Estimates for the mean against the true values from the Gbbs sampler of example 2

k	$\hat{\mu}_k$	μ_k
1	2.8551914	3
2	-1.922274	-2
3	1.5689857	1.5

From the table we see that the estimates are close to the true values. The algorithm was implemented using SAS/IML software and the code is shown in Appendix A3.

3.5 CONCLUSION

The concept of the mixture model, EM algorithm and Gibbs sampler laid out in this chapter form the basic framework for the work that is to follow. Instead of focusing on the Gaussian mixture model, in the next chapter we will go a step further and assume instead that the mixture model is actually made up of a weighted sum of variables from a multinomial distribution. The procedure is similar to that of the Gaussian mixture model and we look at it in detail in the next chapter.

CHAPTER FOUR

THE MULTINOMIAL MIXTURE MODEL

4.1 INTRODUCTION

Since the focus of this research is the Multinomial Mixture model, a detailed study of its theory is presented in this section. We will also see how the EM algorithm and (collapsed) Gibbs sampler are applied to this model in the context of topic models.

4.2 ASSUMPTIONS

In applying the MM model for text clustering purposes the following assumptions are made:

1. Documents follow a bag-of-words representation.

   Each document is regarded as a collection of words which occur with varying frequencies. In essence, the grammatical structure is entirely ignored, thus each document is regarded as a sample and is represented as a high dimensional vector containing the various word frequencies (De Waal, 2010). The corpus can thus be viewed as a $word \times document$ matrix where each $cell_{ij}$ contains the frequency of word i in document j.

2. Each document belongs to one topic.

A topic is regarded as a distribution over the words in the vocabulary set of the corpus (Blei, 2012). In practice, a corpus containing many documents is likely to contain many topics. Each topic is assumed to follow a multinomial distribution over words thus the corpus is modelled as a mixture of multinomials.

3. Documents are formed by a generative process.

   It is assumed that in the formation of a document, a topic is first chosen at random then the words are independently chosen from this topic (Blei, 2004). The finer details of this process are furnished in Section 4.4 to follow.

4.3 TERMINOLOGY AND NOTATION

Table 4.1 is a summary of some of the notation that will be used.

Table 4.1: Notation

D	The number of documents in the corpus. A corpus is a collection of documents, whilst a document is a sequence of words.
T	The number of topics in the corpus. Each topic is assigned a unique label $t \in 1, 2, ..., T$.
V	The size of the vocabulary. The vocabulary is the set of unique words in the corpus.
w	A word is the basic unit of discrete data. Each word is assigned a unique label $w \in 1, 2, ..., V$.
$\vec{d}$	Document d, represented as a $1 \times V$ vector of word frequencies as dictated by the bag-of-words assumption.
N_d	The number of words in document d.
N_d^w	The number of occurrences of word w in document d. Therefore, a document is represented as $\vec{d} = (N_d^1, N_d^2, ..., N_d^V)$.
C	The collection of documents. According to the bag-of-words representation, the corpus is a $document \times word$ matrix with D rows and V columns.
Z_d	Indicates topic assignments. Its form will be specified depending on the context in which it is used.
m_t	The number of documents in topic t.
n_t	The number of words in topic t.
n_t^w	The number of occurrences of word w in topic t.

4.4 GENERATIVE PROCESS

As mentioned previously, in the Multinomial Mixture model, each document is associated with at most one topic and the words in each document are dependent on the topic (Rigouste *et al.*, 2007). The aforementioned generative process can be expanded for the whole corpus as follows:.

1. Sample $\vec{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_T)$ from a T-dimensional Dirichlet distribution with parameters $\vec{\lambda}_\alpha = (\lambda_\alpha, \lambda_\alpha, ..., \lambda_\alpha)$. It follows from the definition of the Dirichlet distribution that $\alpha_t \geq 0$ for all $t = 1, 2, ..., T$ and $\sum_{t=1}^{T} \alpha_t = 1$.

2. For every topic $t = 1, 2, ..., T$, sample $\vec{\beta}_t = \{\beta_{1t}, \beta_{2t}, ..., \beta_{Vt}\}$ from a V-dimensional Dirichlet distribution with parameter $\vec{\lambda}_\beta = (\lambda_\beta, \lambda_\beta, ..., \lambda_\beta)$. Similarly, $\beta_{wt} \geq 0$ for all t and w, $\sum_{w=1}^{V} \beta_{wt} = 1$ where $w = 1, 2, ..., V$ and $t = 1, 2, ..., T$.

3. For each document, $\vec{d}$, $(d = 1, 2, ..., D)$:

   a) Choose a topic $z_d \in \{1, 2, ..., T\}$ from a $Multinomial(\vec{\alpha})$ where $\vec{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_T)$ represents the topic distribution of the corpus. Stated differently, α_t is the probability of a document belonging to topic t for $t = 1, 2, ..., T$.

   b) Choose the word counts, N_d, and independently choose the words, w_d, for document d, where each word is from a multinomial distribution with parameter $\vec{\beta}_t = \{\beta_{1t}, \beta_{2t}, ..., \beta_{Vt}\}$ where $\vec{\beta}_t$ is the word-topic distribution, thus, $\beta_{wt} = P(\text{word } w \text{ belonging to topic } t)$.

Note, the choice of Dirichlet priors is due to its conjugacy to the multinomial distribution.[1]

Graphically, this process can be summarized in a directed model as shown in Figure 4.1.

According to the Multinomial Mixture model proposed by Nigam *et al.* (2000) for text clustering, a document can be modelled as a mixture of multinomials as follows:

$$p(\vec{d}|\vec{\alpha}, \beta) = \sum_{t=1}^{T} \alpha_t \frac{N_d!}{\prod_{w=1}^{V} N_d^w!} \prod_{w=1}^{V} \beta_{wt}^{N_d^w} \tag{4.1}$$

[1] Suppose $Q \sim F$ and $X|q \sim G$. Q is said to be conjugate to X if $Q|X$ follows a distribution in the same family as Q.

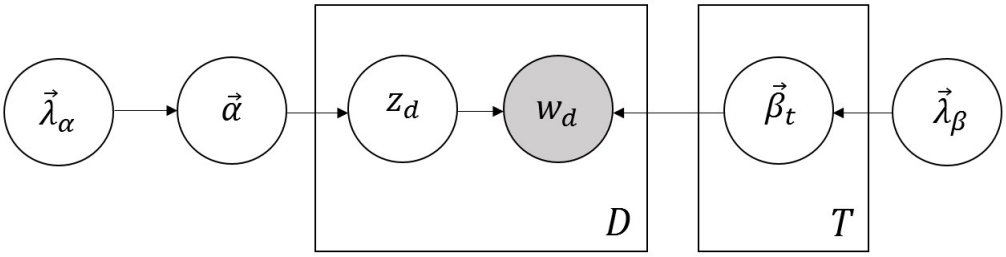


Figure 4.1: Multinomial Mixture graphical model

where $\{\alpha_1, \alpha_2, ..., \alpha_T, \vec{\beta_1}, \vec{\beta_2}, ..., \vec{\beta_T}\}$ denotes the set of model parameters and $\beta = (\vec{\beta_1}, \vec{\beta_2}, ..., \vec{\beta_T}) = (\beta_{wt})_{w=1,2,...,V;t=1,2,...,T}$. In the context of mixture models, the topic distribution, $\vec{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_T)$, is equivalent to the mixing weights.

4.5 ESTIMATION

4.5.1 EM ALGORITHM

In order to estimate the parameters of the model we introduce the latent variable $\vec{z}_d = (z_{d1}, z_{d2}, ..., z_{dT})$ where

$$z_{dt} = \begin{cases} 1 & \text{if document } \vec{d} \text{ is from topic } t \\ 0 & \text{otherwise} \end{cases}$$

for $t \in \{1, ..., T\}$ and $d = 1, 2, ...D$.

The latent indicator vector and the observed document vector form the complete dataset whose joint density function is

$$p(\vec{d}, \vec{z}_d | \vec{\alpha}, \beta) = \prod_{t=1}^{T} \left(\alpha_t \frac{N_d!}{\prod_{w=1}^{V} N_d^w!} \prod_{w=1}^{V} \beta_{wt}^{N_d^w} \right)^{z_{dt}}. \tag{4.2}$$

In the supervised scenario where the documents are labelled, that is $\vec{z}_d$ is known, the unknown parameters can be estimated using the Naive Bayes classifier. Unfortunately, due to the presence of the latent variable, maximizing equation (4.2) is generally intractable. However, due to the existence of the supervised model it is possible to apply the EM algorithm which is an unsupervised

learning technique. The parameters are then estimated in such a way as to maximize the corpus likelihood.

Since each document is independent it follows from equation (4.2) that the log of the corpus likelihood is

$$\begin{aligned} L &= \log \prod_{d=1}^{D} p(\vec{d}, \vec{z}_d | \vec{\alpha}, \beta) \\ &= \log \prod_{d=1}^{D} \prod_{t=1}^{T} \left(\alpha_t \frac{N_d!}{\prod_{w=1}^{V} N_d^w!} \prod_{w=1}^{V} \beta_{wt}^{N_d^w} \right)^{z_{dt}} \\ &\propto \sum_{d=1}^{D} \sum_{t=1}^{T} z_{dt} \left(\log \alpha_t + \sum_{w=1}^{V} N_d^w \log \beta_{wt} \right). \end{aligned} \tag{4.3}$$

Taking $\hat{\vec{\alpha}}$ and $\hat{\vec{\beta}}_t$ to be the current estimates of the parameters, in the expectation step (E-step), z_{dt} is estimated by

$$\theta_{dt} = E(z_{dt} | C, \vec{\alpha}, \beta) = \frac{\hat{\alpha}_t \prod_{w=1}^{V} \hat{\beta}_{wt}^{N_d^w}}{\sum_{t'=1}^{T} \hat{\alpha}_{t'} \prod_{w=1}^{V} \hat{\beta}_{wt'}^{N_d^w}}. \tag{4.4}$$

This is equivalent to calculating the responsibilities as shown in equation (3.1).

$\Theta = (\theta_{dt})_{d=1,\ldots,D;t=1,\ldots,T}$ can be thought of as a *topic probability* matrix with dimensions $D \times T$ whose elements θ_{dt} denote the probability of document d belonging to topic t (Li and Zhang, 2008).

In the maximization step (M-step), the parameters $\{\alpha_t\}$ and $\{\beta_{wt}\}$ are updated:

$$\alpha_t \propto \lambda_\alpha - 1 + \sum_{d=1}^{D} \theta_{dt} \tag{4.5}$$

$$\beta_{wt} \propto \lambda_\beta - 1 + \sum_{d=1}^{D} N_d^w \theta_{dt} \tag{4.6}$$

such that

$$\begin{cases} \sum_{t=1}^{T} \alpha_t = 1 \\ \sum_{w=1}^{V} \beta_{wt} = 1 \end{cases}$$

for $t \in \{1, ..., T\}$.

The Algorithm 5 summarizes this estimation procedure.

Algorithm 5: The EM Algorithm for the T-component Multinomial Mixture

1 Initialise $\vec{\alpha}, \vec{\beta}_t$.

2 *E-step:* determine the responsibilities θ_{dt} for $d = 1, 2, ..., D$ and $t = 1, 2, ..., T$ as in (4.4).

3 *M-step:* compute the parameters $\hat{\vec{\alpha}}$ and $\hat{\vec{\beta}}_t$ $(t = 1, 2, ..., T)$ as in (4.5) and (4.6) respectively.

4 Repeat steps 2 and 3 until the iteration converges.

Since the vocabulary set is very large, many words will have very low probabilities of being in a specific topic. According to Rigouste *et al.* (2007), word probabilities should not get too small, thus the smoothing parameter $\lambda_\beta - 1$ is set to be around the optimal value of 0.2. In addition, since $\lambda_\alpha - 1$ is always negligible (Rigouste *et al.*, 2007), we will set $\lambda_\alpha - 1$ and $\lambda_\beta - 1$ to be equal to 0 and 0.2, respectively, in our implementation of the model.

4.5.2 THE GIBBS SAMPLER

In applying the Gibbs sampler to the Multinomial Mixture model we follow the procedure laid out in Yin and Wang (2014). In the context of Gibbs sampling the Multinomial Mixture is commonly referred to as the *Dirichlet* Multinomial Mixture (DMM)[2]. The Gibbs sampler is a Markov Chain Monte Carlo (MCMC) method whose aim is to construct a Markov Chain whose stationary distribution is the target posterior distribution (Darling, 2011).

The core problem is that of posterior inference. In view of the generative process shown in Figure 4.1, the task of Bayesian inference is to *generate* parameter values from given observations whilst handling the complications that arise with the presence of hidden values in the model. In essence, this can be thought of as *inverting* the generative model (Heinrich, 2005).

In connection with the Gibbs sampler, the definition of the latent variable, z, will be slightly modified in order to ease the description of the theory. Instead of letting the variable z be an indicator variable as with the EM algorithm, let the set of topic labels for each document be

$$\vec{z} = \{z_1, z_2, ..., z_D\}$$

[2]The Gibbs sampler is a full Bayesian approach, hence the emphasis on the (Dirichlet) prior in the name, *Dirichlet* Multinomial Mixture.

where $z_d = t \in \{1, 2, ..., T\}$ denotes the topic label of document d.

For the DMM, the parameters of interest are the topic proportions, $\vec{\alpha}$, the topic-word distributions, $\vec{\beta}_t$, and the topic index assignments for each document, $\vec{z}$. As with the EM algorithm, $\vec{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_T)$ represents the topic distribution of the corpus and $\vec{\beta}_t$ is the word-topic distribution where $\beta = (\vec{\beta}_1, \vec{\beta}_2, ..., \vec{\beta}_T) = (\beta_{wt})_{w=1,2,...,V;t=1,2,...,T}$.

According to the generative process described in Section 4.4, the joint distribution is:

$$p(\vec{\alpha}, \beta, \vec{z}, \vec{d}|\vec{\lambda_\alpha}, \vec{\lambda_\beta}) = p(\vec{\alpha}|\vec{\lambda}_\alpha)p(\beta|\vec{\lambda}_\beta)p(\vec{z}|\vec{\alpha})p(\vec{d}|z_d).$$

Whilst conditional distributions can be derived for all the variables, since $\vec{z}$ is a sufficient statistic for $\vec{\alpha}$ and β, they can both be calculated from $\vec{z}$. Consequently, if the parameters $\vec{\alpha}$ and β are integrated out, we simply need to sample from $\vec{z}$. This is referred to as the *uncollapsed* Gibbs sampler (Darling, 2011). Stated differently, instead of obtaining a sample from $p(\alpha, \beta, z|\lambda_\alpha, \lambda_\beta)$ we only have to sample from

$$p(z_d|\vec{z}_{\neg d}, C, \vec{\lambda}_\alpha, \vec{\lambda}_\beta),$$

where $\vec{z}_{\neg d}$ denotes all the topics except for z_d and C denotes the corpus. We will continue to use $\neg d$ to indicate when the d-th document is omitted.

In deriving this distribution we follow Yin and Wang (2014) with the help of Heinrich (2005) to furnish the missing details. From the rules of conditional probability it follows that

$$\begin{aligned} p(z_d|\vec{z}_{\neg d}, C, \vec{\lambda}_\alpha, \vec{\lambda}_\beta) &= \frac{p(\vec{z}, C|\vec{\lambda}_\alpha, \vec{\lambda}_\beta)}{p(\vec{z}_{\neg d}, C|\vec{\lambda}_\alpha, \vec{\lambda}_\beta)} \quad \text{(since } \vec{z}_{\neg d} \text{ and } z_d \text{ is just } \vec{z}\text{)} \\ &\propto \frac{p(\vec{z}, C|\vec{\lambda}_\alpha, \vec{\lambda}_\beta)}{p(\vec{z}_{\neg d}, C_{\neg d}|\vec{\lambda}_\alpha, \vec{\lambda}_\beta)}. \end{aligned} \tag{4.7}$$

where $C_{\neg d}$ is the corpus excluding document d.

In order to find the form of expression (4.7), we first derive an expression for the numerator, and then adapt it for the denominator, which will have a similar form, except that document d is omitted.

From the graphical model in Figure 4.1 it can be seen that

$$p(\vec{z}, C|\vec{\lambda}_\alpha, \vec{\lambda}_\beta) = p(\vec{z}|\lambda_\alpha)p(C|\vec{z}, \lambda_\beta). \tag{4.8}$$

The first term is independent of λ_β, whilst the second is independent of λ_α, thus the two parts are handled separately. Beginning with the first term of equation (4.8), it can be derived by integrating with respect to α. As described in the generative process,

$$p(\vec{z}|\vec{\alpha}) = \prod_{d=1}^{D} p(z_d = t|\vec{\alpha}) = \prod_{d=1}^{D} \alpha_{z_d} = \prod_{t=1}^{T} \alpha_t^{m_t}$$

follows a multinomial distribution[3] and $p(\vec{\alpha}|\vec{\lambda}_\alpha)$ follows a Dirichlet distribution. Thus,

$$\begin{aligned} p(\vec{z}|\vec{\lambda}_\alpha) &= \int p(\vec{z}|\vec{\alpha})p(\vec{\alpha}|\vec{\lambda}_\alpha)\mathrm{d}\vec{\alpha} \\ &= \int \left(\prod_{t=1}^{T} \alpha_t^{m_t}\right)\left(\frac{1}{\Delta(\vec{\lambda}_\alpha)}\prod_{t=1}^{T} \alpha_t^{\lambda_\alpha - 1}\mathrm{d}\vec{\alpha}\right) \\ &= \frac{1}{\Delta(\vec{\lambda}_\alpha)}\int \prod_{t=1}^{T} \alpha_t^{m_t+\lambda_\alpha-1}\mathrm{d}\vec{\alpha} \\ &= \frac{\Delta(\vec{m} + \vec{\lambda}_\alpha)}{\Delta(\vec{\lambda}_\alpha)}, \quad \vec{m} = \{m_t\}_{t=1}^{T}, \end{aligned} \tag{4.9}$$

where m_t is the number of documents in topic t. Adopting the delta function notation as in Heinrich (2005) we have that

$$\Delta(\vec{\lambda}_\alpha) = \frac{\prod_{t=1}^{T}\Gamma(\lambda_\alpha)}{\Gamma(\sum_{t=1}^{T}\lambda_\alpha)}$$

and

$$\Delta(\vec{m} + \lambda_\alpha) = \frac{\prod_{t=1}^{T}\Gamma(m_t + \lambda_\alpha)}{\Gamma(\sum_{t=1}^{T} m_t + \lambda_\alpha)} = \frac{\prod_{t=1}^{T}\Gamma(m_t + \lambda_\alpha)}{\Gamma(D + T\lambda_\alpha)}$$

where Γ denotes the gamma function, and $D = \sum_{t=1}^{T} m_t$, the number of documents in the dataset. The last step in equation (4.9) follows from integrating the Dirichlet distribution function over its

[3] In line with the bag of words assumption, the order of the words is not important, thus the multinomial coefficient is dropped (Heinrich, 2005).

support:

$$\int f(\vec{p}|\vec{\alpha})\mathrm{d}\vec{p} = \int \frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}p_k^{\alpha_k-1}\mathrm{d}\vec{p} = 1.$$

The second term of equation (4.8) is found similarly, by integrating with respect to β. As described in the generative process, $p(\vec{\beta}|\vec{\lambda}_\beta)$ follows a Dirichlet distribution and $p(C|\vec{z},\vec{\beta})$ assuming words are generated independently from a multinomial distribution (topic) it follows that:

$$\begin{aligned} p(C|\vec{z},\vec{\lambda}_\beta) &= \int p(C|\vec{z},\vec{\beta})p(\vec{\beta}|\vec{\lambda_\beta})\mathrm{d}\beta_t \\ &= \int \left(\prod_{t=1}^{T}\prod_{w=1}^{V}\beta_{wt}^{n_t^w}\right)\left(\frac{1}{\Delta(\vec{\beta})}\prod_{w=1}^{V}\beta_{wt}^{\lambda_\beta-1}\right)\mathrm{d}\beta_t \\ &= \prod_{t=1}^{T}\frac{1}{\Delta(\vec{\lambda}_\beta)}\int\prod_{w=1}^{V}\beta_{wt}^{n_t^w+\lambda_\beta-1}\mathrm{d}\beta \\ &= \prod_{t=1}^{T}\frac{\Delta(\vec{n}_t+\vec{\lambda}_\beta)}{\Delta(\vec{\lambda}_\beta)}, \quad \vec{n}_t = \{n_t^w\}_{w=1}^{V}, \end{aligned} \tag{4.10}$$

where n_t^w is the number of occurrences of word w in topic t. Similarly,

$$\Delta(\vec{\lambda}_\beta) = \frac{\prod_{m=1}^{V}\Gamma(\lambda_\beta)}{\Gamma(\sum_{m=1}^{V}\lambda_\beta)}$$

and

$$\Delta(\vec{n}_t+\vec{\lambda}_\alpha) = \frac{\prod_{w=1}^{V}\Gamma(n_t^w+\lambda_\beta)}{\Gamma(\sum_{w=1}^{V}\vec{n}_t^w+\lambda_\beta)} = \frac{\prod_{w=1}^{V}\Gamma(n_t^w+\lambda_\beta)}{\Gamma(n_t+V\lambda_\beta)}$$

where n_t is the number of words in topic t and $n_t = \sum_{w=1} n_t^w$.

From equations (4.9) and (4.10) The joint distribution in equation (4.8) then becomes

$$p(\vec{z},C|\lambda_\alpha,\lambda_\beta) = \frac{\Delta(\vec{m}+\vec{\lambda}_\alpha)}{\Delta(\vec{\lambda}_\alpha)}\prod_{t=1}^{T}\frac{\Delta(\vec{n}_t+\vec{\lambda}_\beta)}{\Delta(\vec{\lambda}_\beta)}. \tag{4.11}$$

Using equation (4.11), equation (4.7) can now be expressed as follows:

$$
\begin{aligned}
p(z_d|\vec{z}_{\neg d}, C) &\propto \frac{p(\vec{z}, C|\vec{\lambda}_\alpha, \vec{\lambda}_\beta)}{p(\vec{z}_{\neg d}, C_{\neg d}|\vec{\lambda}_\alpha, \vec{\lambda}_\beta)} \\
&\propto \frac{\Delta(\vec{m} + \vec{\lambda}_\alpha)}{\Delta(\vec{m}_{\neg d} + \vec{\lambda}_\alpha)} \frac{\Delta(\vec{n}_t + \vec{\lambda}_\beta)}{\Delta(\vec{n}_{t,\neg d} + \vec{\lambda}_\beta)} \\
&= \frac{\prod_{t=1}^{T} \Gamma(m_t + \lambda_\alpha)/\Gamma(D + T\lambda_\alpha)}{\prod_{t=1}^{T} \Gamma(m_{t,\neg d} + \lambda_\alpha)/\Gamma(D - 1 + T\lambda_\alpha)} \frac{\prod_{w=1}^{V} \Gamma(n_t^w + \lambda_\beta)/\Gamma(n_t + V\lambda_\beta)}{\prod_{w=1}^{V} \Gamma(n_{t,\neg d}^w + \lambda_\beta)/\Gamma(n_{t,\neg d} + V\lambda_\beta)} \\
&= \frac{\prod_{t=1}^{T} \Gamma(m_t + \lambda_\alpha)}{\Gamma(D + T\lambda_\alpha)} \frac{\Gamma(D - 1 + T\lambda_\alpha)}{\prod_{t=1}^{T} \Gamma(m_{t,\neg d} + \lambda_\alpha)} \frac{\prod_{w=1}^{V} \Gamma(n_t^w + \lambda_\beta)}{\Gamma(n_t + V\lambda_\beta)} \frac{\Gamma(n_{t,\neg d} + V\lambda_\beta)}{\prod_{w=1}^{V} \Gamma(n_{t,\neg d}^w + \lambda_\beta)} \\
&\propto \frac{\Gamma(m_t + \lambda_\alpha)}{\Gamma(m_{t,\neg d} + \lambda_\alpha)} \frac{\Gamma(D - 1 + T\lambda_\alpha)}{\Gamma(D + T\lambda_\alpha)} \frac{\prod_{w \in d} \Gamma(n_t^w + \lambda_\beta)}{\prod_{w \in d} \Gamma(n_{t,\neg d}^w + \lambda_\beta)} \frac{\Gamma(n_{t,\neg d} + V\lambda_\beta)}{\Gamma(n_t + V\lambda_\beta)}.
\end{aligned}
\tag{4.12}
$$

Since $m_t = m_{t,\neg d} + 1$ and $n_t = n_{t,\neg d} + N_d$, where N_d denotes the number of words in document d, the first two terms of (4.12) can be simplified as follows:

$$
\frac{\Gamma(m_t + \lambda_\alpha)}{\Gamma(m_{t,\neg d} + \lambda_\alpha)} = \frac{\Gamma(m_{t,\neg d} + 1 + \lambda_\alpha)}{\Gamma(m_{t,\neg d} + \lambda_\alpha)} = \frac{(m_{t,\neg d} + \lambda_\alpha)\Gamma(m_{t,\neg d} + \lambda_\alpha)}{\Gamma(m_{t,\neg d} - 1 + \lambda_\alpha)} = m_{t,\neg d} + \lambda_\alpha
$$

and

$$
\frac{\Gamma(D - 1 + T\lambda_\alpha)}{\Gamma(D + T\lambda_\alpha)} = \frac{\Gamma(D - 1 + T\lambda_\alpha)}{(D - 1 + T\lambda_\alpha)\Gamma(D - 1 + T\lambda_\alpha)} = \frac{1}{D - 1 + T\lambda_\alpha}.
$$

Also, since $\frac{\Gamma(x+m)}{\Gamma(x)} = \prod_{i=1}^{m}(x + i - 1)$, it follows that the last term of (4.12) can be expressed as

$$
\frac{\Gamma(n_t + V\lambda_\beta)}{\Gamma(n_{t,\neg d} + V\lambda_\beta)} = \frac{\Gamma(n_{t,\neg d} + V\lambda_\beta + N_d)}{\Gamma(n_{t,\neg d} + V\lambda_\beta)} = \prod_{i=1}^{N_d}(n_{t,\neg d} + V\lambda_\beta + i - 1).
$$

Substituting these simplifications, it therefore follows that (4.12) can be written as

$$p(z_d|\vec{z}_{\neg d}, C) \propto \frac{m_{t,\neg d} + \lambda_\alpha}{D - 1 + T\lambda_\alpha} \frac{\frac{\prod_{w \in d} \Gamma(n_t^w + \lambda_\beta)}{\prod_{w \in d} \Gamma(n_{t,\neg d}^w + \lambda_\beta)}}{\prod_{i=1}^{N_d} (n_{t,\neg d} + V\lambda_\beta + i - 1)}. \tag{4.13}$$

If we assume that a word can only appear at most once within a document then

$$\frac{\prod_{w \in d} \Gamma(n_t^w + \lambda_\beta)}{\prod_{w \in d} \Gamma(n_{t,\neg d}^w + \lambda_\beta)} = \frac{\prod_{w \in d} \Gamma(n_{t,\neg d}^w + \lambda_\beta + 1)}{\prod_{w \in d} \Gamma(n_{t,\neg d}^w + \lambda_\beta)} = \prod_{w \in d} \Gamma(n_{t,\neg d}^w + \lambda_\beta)$$

since $n_t^w = n_{t,\neg d}^w + 1$ and $\frac{\Gamma(x+m)}{\Gamma(x)} = \prod_{i=1}^{m}(x + i - 1)$.

Allowing a word to occur more than once in a document results in

$$\frac{\prod_{w \in d} \Gamma(n_t^w + \lambda_\beta)}{\prod_{w \in d} \Gamma(n_{t,\neg d}^w + \lambda_\beta)} = \prod_{w \in d} \prod_{j=1}^{N_d^w} \Gamma(n_{t,\neg d}^w + \lambda_\beta + j - 1).$$

since $n_t^w = n_{t,\neg d}^w + N_d^w$ where N_d^w is the number if occurrences of words w in document d.

From this, equation (4.13) can be expressed as

$$p(z_d|\vec{z}_{\neg d}, C) \propto \frac{m_{t,\neg d} + \lambda_\alpha}{m_{t,\neg d} + \lambda_\alpha} \frac{\prod_{w \in d} \prod_{j=1}^{N_d^w} \Gamma(n_{t,\neg d}^w + \lambda_\beta + j - 1)}{\prod_{i=1}^{N_d} (n_{t,\neg d} + V\lambda_\beta + i - 1)}. \tag{4.14}$$

ESTIMATING THE TOPIC DISTRIBUTION $\vec{\alpha}$ AND WORD-TOPIC DISTRIBUTION β

As mentioned before, once we have sampled topic assignments, z_d, for each document in the Gibbs sampler, we can calculate the topic distribution, α, and the word-topic distribution, β.

Since the Dirichlet distribution is conjugate to the multinomial distribution, it follows that:

$$p(\vec{\beta}_t|C, \vec{z}, \vec{\lambda}_\beta) = Dir(\vec{\beta}_t|\vec{n}_t + \vec{\lambda}_\beta)$$

where $\vec{n}_t = \{n_t^w\}_{w=1}^V$ and n_t^w is the number of occurrences of word w in topic t.[4] Consequently,

[4]Due to the conjugacy of the Dirichlet distribution to the multinomial distribution, it follows that: if $Q \sim Dir(\vec{\alpha})$ and $X|q \sim Mulinomial(\vec{q})$ where $\vec{n}$ is the number of successes of each of the trials, then the $Q|x \sim Dir(\vec{n} + \vec{\alpha})$.

from the posterior mean[5] of β,

$$\beta_{wt} = \frac{n_t^w + \lambda_\beta}{\sum_{w=1}^{V} n_t^w + V\lambda_\beta}.$$

Similarly, $\vec{\alpha}$ since $p(\vec{\alpha}_t|C, \vec{z}, \vec{\lambda}_\alpha) = Dir(\vec{\beta}_t|\vec{m} + \vec{\lambda}_\alpha)$ it follows that

$$\alpha_t = \frac{m_t + \lambda_\alpha}{\sum_{t=1}^{T} m_t + T\lambda_\alpha}.$$

THE GIBBS SAMPLING ALGORITHM FOR THE DIRICHLET MULTINOMIAL MIXTURE MODEL

In the collapsed Gibbs sampler, the first step is to randomly assign topic labels to each document by sampling the topic assignments, $z_d \sim Multinomial(1/K)$, where K is the number of topics. From these assignments the values of, m_t, (the number of documents in topic t), n_t (the number of words in topic t) and n_t^w (the number of occurrences of word w in cluster t) are calculated. We then take a document, remove its original topic index from $\vec{z}$ and reassign it to a new topic by sampling from equation (4.14). This, in turn, involves updating m_t, n_t and n_t^w accordingly, and the process is alternately repeated for each document until the sample converges. According to Yin and Wang (2014), the Gibbs sampler is very efficient and can achieve good and stable performance in just $I = 5$ iterations.

As in Yin and Wang (2014), the algorithm required to implement the Gibbs sampler is shown in Algorithm 6.

[5]The mean of a Dirichlet distribution with parameter $\vec{\alpha}$ is $\frac{\alpha_k}{\sum_k \alpha_k}$.

Algorithm 6: The Gibbs Sampling Algorithm for the Dirichlet Multinomial Mixture model

Data: Documents in the input, C
Result: Topic labels, $\vec{z}$, for each document

begin
  initialize m_t, n_t and n_t^w as zero for each topic t
  for *each document* $d \in [1, D]$ **do**
    sample a topic for $d : z_d \sim Multinomial(1/T)$
    $m_t \leftarrow m_t + 1$ and $n_t \leftarrow n_t + N_d$
    for *each word* $w \in d$ **do**
      $n_t^w \leftarrow n_t^w + N_d^w$
  for $i \in [1, I]$ **do**
    record the current topic of $d : t = z_d$
    $m_t \leftarrow m_t - 1$ and $n_t \leftarrow n_t - N_d$
    for *each word* $w \in d$ **do**
      $n_t^w \leftarrow n_t^w - N_d^w$
    sample a topic for d: $z_d \sim p(z_d|\vec{z}_{\neg d}, C)$ (equation (4.14))
    $m_t \leftarrow m_t + 1$ and $n_t \leftarrow n_t + N_d$
    for *each word* $w \in d$ **do**
      $n_t^w \leftarrow n_t^w + N_d^w$
end

INTERPRETATION OF THE DIRICHLET PARAMETERS

In the article titled *A Dirichlet Multinomial Mixture Model-based Approach for Short Text Clustering* by Yin and Wang (2014), the meaning of the Dirichlet parameters as well their theoretic influence is discussed:

The Dirichlet parameter, λ_α, relates the prior probability of a document being assigned to a topic. If $\lambda_\alpha = 0$, once the number of documents assigned to a topic becomes zero, a document will never be assigned to that topic. This is due to the fact that the first part of equation (4.14) becomes zero. If the value of λ_α is increased, then the probability of a document being assigned to a topic also increases.

The other Dirichlet parameter, λ_β, influences how a document is assigned to a topic according to thematic similarity with other documents. If λ_β is set to zero, a document will not be assigned to a topic if a word in the document is not present in the topic. This is not sensible as other words in the document may appear many times in the topic and the document may be thematically similar

to other documents in the topic.

The DMM model assumes symmetric Dirichlet priors. This means that all topics have the same λ_α's and the same λ_β's which implies that all topics are initially given equal importance and all words are of equal importance, respectively. The former implication ties in with our intuition, but the latter is not as sensible. Words that occur in all documents do not provide any information that would be useful in discriminating between different documents. Taking this into account, it would therefore be sensible to put less emphasis on words that are too popular by assigning larger λ_β values to such words (Yin and Wang, 2014).

The investigation of the potential for incorporating global weighting for words is an area of future work that Yin and Wang (2014) have expressed an interest in exploring. In our experiments we set both λ_α and λ_β to 0.1 as in Yin and Wang (2014).

4.6 CONCLUSION

Below is a graph showing the results of log-likelihood calculated from a small example $document \times word$ matrix (see Appendix B1). The value of the log-likelihood is calculated at each of the 50 iterations. This is repeated 50 times for both estimation methods: the EM algorithm and Gibbs sampler. The estimation algorithms are both implemented in *Python* and shown in Appendix B2 and B3 respectively.

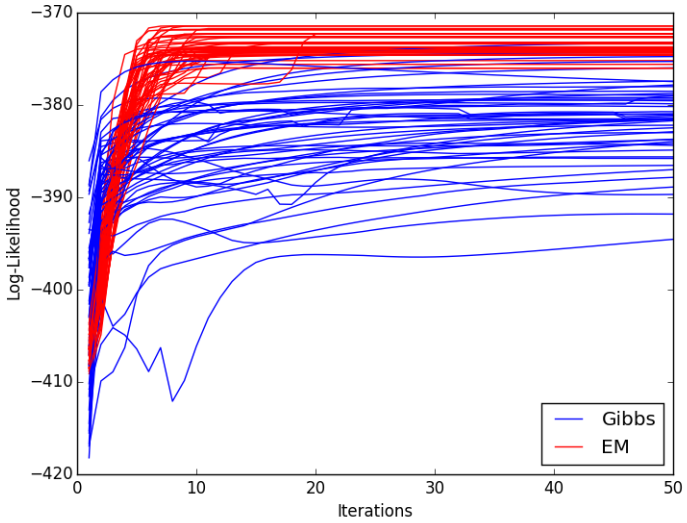


Figure 4.2: Log-likelihood of example $document \times word$ matrix

Form Figure 4.2, we see that the log-likelihoods for the model inferred from the Gibbs sampler display a greater variance than those inferred by the EM algorithm. However, despite this, the convergence of the Gibbs sampler appears to be sooner for most of the iterations. It can also be seen that the EM converges to higher values then the Gibbs, however, given the scale of the graph, the difference is relatively small.

In our experiments, we chose to use the Gibbs sampler due to its faster convergence. According to Yin and Wang (2014), the Gibbs sampler should converge in just $I = 5$ iterations and, from Figure 4.2, this appears to be supported. Another reason for this decision, is the fact that the implementation of the LDA that we had access to is also estimated using the Gibbs sampler. Therefore, in light of our choice to implement the Gibbs sampler, we will often refer to the model as the Dirichlet Multinomial Mixture (DMM) model.

CHAPTER FIVE

EVALUATION METHODS

5.1 INTRODUCTION

Due to the unsupervised nature of the topic models we are implementing, model selection is not a trivial task. Topic models may be applied with the intention of classifying documents according to thematic similarity or for the purposes of information retrieval (Wallach *et al.*, 2009). Some measures are only useful for evaluating the predictive capabilities of the model, whilst others are more useful in helping to assess its exploratory ability (Chang *et al.*, 2009). In this chapter, we present some of the various evaluation measures that have been proposed.

5.2 PERPLEXITY

One of the most common measures used to assess the generalization ability of a model is perplexity. Perplexity is calculated based on the likelihood of a test set of documents, given some training documents. According to Heinrich (2005), it is defined as the "reciprocal geometric mean of the likelihood of a test corpus given the model or its equivalent". Assuming that the test set, C^*, contains D^* documents each of length N_d then perplexity is calculated as follows:

$$Perplexity(C^*) = \exp\left\{-\frac{\sum_{d=1}^{D^*} \log p(\vec{d})}{\sum_{d=1}^{D^*} N_d}\right\}$$

where $\log p(\vec{d})$ denotes the log-likelihood of a document. The normalization by the total number of word occurrences makes it possible for perplexity to be used to compare different models (Rigouste *et al.*, 2007). Since a high likelihood on the test set is desirable, it follows from the above equation that a low perplexity is desirable. Consequently, the lower the perplexity values, the better the words of the test set are represented by the words of the trained topics, hence the better the generalizing ability of the model. Conversely, a high perplexity is indicative of a poor representation of the words of the test set by the words of the trained topics (Heinrich, 2005), hence a poor ability to predict the topic structure of new documents.

One of the weaknesses of using perplexity as a performance measure however, is that it tends to have a high variance. In addition, according to Chang *et al.* (2009), models which may have a desirably low perplexity typically produce topics which are less interpretable. They propose two measures that quantify the interpretability of a topic model, namely *word intrusion* and *topic intrusion*, which will be discussed further below.

According to Wallach *et al.* (2009) many models that try to estimate the probability of a held-out dataset given a trained model are often inaccurate. The reader is referred to Wallach *et al.* (2009) where two supposedly more accurate and efficient methods are presented, namely *Chibb-style estimation* and the *left-to-right evaluation algorithm*. In general, although methods of estimating the probability of a held out dataset exist, it still remains a major issue. We will not discuss these methods further here as they lie beyond the scope of this work.

5.3 WORD INTRUSION

Chang *et al.* (2009) noted that most topic modelling papers present qualitative assessments of inferred topics or merely make the assumption that the topics proposed are coherent. They propose a measure called *word intrusion* which checks whether inferred topics are analogous with natural word groupings that are sensible to humans. In other words, it allows the user to assess whether topics are semantically *cohesive*.

In order to implement the word intrusion task, a subject is given six randomly ordered words from which the word that does not belong must be identified. For instance, using the example of Chang *et al.* (2009), in the set $\{$*dog, apple, cat, horse, pig, cow*$\}$, subjects usually identified *apple* as the "intruder" since the remaining words make logical sense. On the other hand, given the set

{*car, teacher, platypus, agile, blue, Zaire*}, subjects would typically make a random selection as the intruder, implying that the topic coherence is low.

The set of words is created by taking the five words with the highest probabilities in a topic and including an intruder which is chosen to be a word that has a low probability in the topic, but a high probability in another topic. The words are then presented to the subject after they have been randomly ordered (Chang *et al.*, 2009).

The topic coherence, as measured by word intrusion, is calculated as the fraction of subjects who agree with the model. The measure is called *model precision* and is calculated as

$$\text{MP}_k^m = \sum_s \mathbb{I}(i_{k,s}^m = w_k^m)/S.$$

$\mathbb{I}(x_1 = x_2)$ is 1 if $x_1 = x_2$ and 0 otherwise, whilst w_k^m denotes the index of the intruding word in topic k inferred by model m, $i_{k,s}^m$ denotes the index of the intruding word in topic k inferred by model m selected by subject s and S is the total number of subjects. The higher the model precision, the better the performance of the model with regards to uncovering semantically coherent topics (Chang *et al.*, 2009).

One logical disadvantage of this measure is that due to its human-based nature it would be difficult to estimate the variance of the statistic. Given a different set of subjects, one may get a different model precision and the intruder selection may be influenced by a subject's knowledge.

5.4 TOPIC INTRUSION

Topic intrusion is another measure proposed by Chang *et al.* (2009) which aims to assess the extent to which the topics of a document that are uncovered by the model agree with human perception of the topical content of the document.

In this task, the subject is presented with a short extract from the document and its title. The subject is then shown a list of 4 topics, represented by the top 8 words of the topic as inferred by the model. Three of the topics are the topics to which the model assigns the highest probability for the document whilst the fourth is randomly selected from the topics that have a low probability according to the model. Similar to the word intrusion task, the subject than selects the intruder topic. Subjects are only presented with a snippet of the document in order to save time. According to Chang *et al.* (2009), although subjects may not be presented with the full document as the model

is, humans are very good at extrapolating from limited data. As before, if subjects continually disagree with the model about which topic is the intruder, this implies that the model failed to infer topic assignments that are relevant and intuitive compared to human perception.

Word intrusion is quantified by a statistic called the *topic log odds*. It measures the extent to which the model and human judgement agree. Chang *et al.* (2009) defined it as the log of the ratio of the probability mass assigned to the true intruder to the probability mass assigned to the subject-selected intruder and is calculated as,

$$\text{TLO}_d^m = (\sum_s \log \hat{\theta}^m_{d,j^m_{d,*}} - \log \hat{\theta}^m_{d,j^m_{d,s}})/S.$$

$\hat{\theta}^m_d$ denotes the point estimate of the topic proportions vector assigned to document d by model m. $j^m_{d,s} \in \{1, 2, ..., K\}$ denotes the intruder selected by subject s whilst $j^m_{d,*}$ denotes the "true" intruder according to the model. The higher the value of the topic log odds the better the topic proportion inferred by the model corresponds with human judgement. (Chang *et al.*, 2009)

As with word intrusion, this model allows the user to assess the quality of the inferred topics in relation to human logic, but it suffers from the same weakness as word intrusion due the its basis on human judgement.

5.5 VARIATION OF INFORMATION

Variation of information is measure that was proposed by Meilă (2003). The objective of variation of information (VI) is to compare two clusterings by measuring the amount of information gained or lost when switching from one clustering, $\mathcal{C}$, to another clustering, $\mathcal{C}'$. In the context of topic models, the clusterings would be analogous to the classification of documents to topics according to thematic similarity where $\mathcal{C}$ and $\mathcal{C}'$ represent clusterings created by two different topic models. Following Meilă (2003), this measure can be described as follows:

Assuming that each observation has an equal probability of being allocated to a cluster, the probability of an observation being assigned to cluster $\mathcal{C}_k$ equals

$$P(k) = \frac{n_k}{n}.$$

This defines a discrete random variable associated with clustering $\mathcal{C}$ that takes on K values.

The uncertainty of the allocation is equivalent to the entropy of this random variable and is defined as

$$H(\mathcal{C}) = -\sum_{k=1}^{K} P(k) \log P(k). \tag{5.1}$$

Let $P(k)$, $k = 1, 2, ..., K$ denote the random variable associated with clustering $\mathcal{C}$, whilst $P'(k')$, $k' = 1, 2, ..., K'$ denotes the random variable associated with clustering $\mathcal{C}'$. Thus, the joint distribution of the two random variables, $P(k, k')$, represents the probability that an observation belongs to clustering $\mathcal{C}_k$ in clustering $\mathcal{C}$ and to clustering $\mathcal{C}'_{k'}$ in clustering $\mathcal{C}'$.

$$P(k, k') = \frac{|\mathcal{C}_k \cap \mathcal{C}'_{k'}|}{n}.$$

Given the aforementioned marginal and joint distributions, the *mutual information* between clusterings $\mathcal{C}$ and $\mathcal{C}'$ is equal to the mutual information between the two clusterings which is defined as

$$I(\mathcal{C}, \mathcal{C}') = \sum_{k=1}^{K} \sum_{k'=1}^{K'} P(k, k') \log \frac{P(k, k')}{P(k)P'(k')}. \tag{5.2}$$

$I(\mathcal{C}, \mathcal{C}')$ can be thought of as follows: Suppose a random observation is assigned to a cluster in $\mathcal{C}'$. The uncertainty associated with this allocation is $H(\mathcal{C}')$. If it then becomes known as to which cluster this observation belongs in clustering $\mathcal{C}$, then the reduction in the uncertainty about $\mathcal{C}'$ is $I(\mathcal{C}, \mathcal{C}')$ averaged over all the observations.

Given the entropies for both clusterings, from equation (5.1), and the mutual information, from equation (5.2), VI is calculated as

$$VI(\mathcal{C}, \mathcal{C}') = H(\mathcal{C}) + H(\mathcal{C}') - 2I(\mathcal{C}, \mathcal{C}').$$

The lower the VI, the more similar are the two clusterings that are being compared.

VI has many positive characteristics that include the fact that it does not make any assumptions about how a clustering was formed and it is applicable to both soft (probabilistic) and hard (non-probabilistic) clustering (Meilă, 2003). In addition, it is a true metric as it is always non-negative, equalling zero when $\mathcal{C}$ and $\mathcal{C}'$ are identical, it is symmetric and it obeys the triangle inequality

(Meilă, 2003; Heinrich, 2005).

5.6 COHERENCE

One disadvantage of many evaluation methods is that they are often ad hoc and application specific (Stevens *et al.*, 2012). Some performance measures, like perplexity, aim to evaluate the amount of information captured by the topics, whilst others, such as word and topic intrusion, focus on comparing topics to human judgement. In order to simultaneously address these evaluation objectives Stevens *et al.* (2012) devises a generalized coherence metric based on topic coherence measures proposed by Mimno *et al.* (2011) and Newman *et al.* (2010).

Both models compute the coherence of a topic as a sum of similarity scores,

$$coherence(V) = \sum_{v_i, v_j \in V} score(v_i, v_j, \epsilon),$$

where V is the set of words associated with the topic and ϵ is a smoothing power that is included to ensure that the model produces real values.

The UCI metric (Mimno *et al.*, 2011) defines the score as

$$score(v_i, v_j, \epsilon) = \log \frac{p(v_i, v_j) + \epsilon}{p(v_i)p(v_j)} \tag{5.3}$$

where $p(v_i)$ and $p(v_j)$ denotes the probability of the i-th and j-th words and $p(v_i, v_j)$ are their joint distributions. Without the smoothing parameter, ϵ, $score(v_i, v_j)$ is called the pointwise mutual information (PMI) between words v_i and v_j. According to Mimno *et al.* (2011), the logic behind this measure is that if a set of words are coherent, then each word should be associated with all or most of the other words in the set. This association is measured using the PMI.

The UMass metric (Newman *et al.*, 2010) is based on the number of documents with co-occurring words and defines the score as

$$score(v_i, v_j, \epsilon) = \log \frac{D(v_i, v_j) + \epsilon}{D(v_j)} \tag{5.4}$$

where $D(v_i, v_j)$ is the number of documents that contain both words v_i and v_j whilst $D(v_j)$

denotes the number of documents containing word v_j. This model is based on the idea that pairs of words from the same topic will both occur in a document, but pairs of words from different topics will not. In addition, if a topic contains random, unrelated words, then the words of the topic are unlikely to co-occur.

To evaluate a complete model rather than individual topics, Stevens *et al.* (2012) calculates the average of the coherence scores of all the topics as well as the entropy, which is found by omitting the $\log$ and ϵ form the scoring functions given in equations (5.3) and (5.4). The average coherence measures the quality of the model and the entropy provides an indication of whether topics are of approximately uniform quality or if there is a large difference between low and high quality topics (Stevens *et al.*, 2012). A model is considered to perform better in relation to another model if it has a higher average coherence. A low entropy is indicative of the model having learned topics of approximately uniform quality, whereas a high entropy indicates that the model has learned topics with highly differing quality levels. A topic of good quality is regarded as one having related words that are sensible together in defining the topic (Stevens *et al.*, 2012). According to Stevens *et al.* (2012), the choice of whether topics should be more homogeneous in quality or not depends on user preference.

According to Stevens *et al.* (2012), both scoring measures have been shown to agree with human judgements and "provide a balance between internal measures of information gain and comparisons to human ratings of coherence."

5.7 CONCLUSION

This chapter presented metrics designed to evaluate models based on their generalization and classification abilities as well as the quality of the inferred topics. The evaluation of topic models is an area of much debate and there is no consensus regarding which is the best measure to use.

Due the wider spectrum of evaluation covered by coherence, we will use average coherence and entropy based on the UMass metric as the performance measure in our experiments and take $\epsilon = 1$ as was done in the initial article by Mimno *et al.* (2011).

CHAPTER SIX

EXPERIMENTS

6.1 INTRODUCTION

In this section we investigate the validity of our hypothesis that the MM model will outperform the LDA model. Both models are applied to two short and two long text corpora and their performance is assessed based on coherence and entropy. As was mentioned in Chapter 5, these measures are based on scoring measures that have been shown to align with human evaluation and provide a good trade off between assessing the informational content of the model and its ability to produce topics that coincide with human judgement (Stevens *et al.*, 2012). We will also investigate the convergence properties of the two algorithms using perplexity.

In order to gain a clear understanding of the differences and similarities between the LDA and MM model Table 6.1 shows a comparison of the two models.

Table 6.1: Comparison of LDA and MM model

Property	Multinomial Mixture model	Latent Dirichlet Allocation
Document formation	Each document is formed probabilistically through a generative process.	Each document is formed probabilistically through a generative process.
Generative process	For each document select a topic, then select words from that topic.	For each document select a distribution of topics, then select words from the topics.
Number of topics per document	At most one.	More than one topic.
Estimation algorithms	- EM algorithm - Gibbs sampler	- Variational Bayes - Gibbs sampler - Laplace Approximation
Document Probability	Equation (4.1)	Equation (2.1)

From the graphical models, it can also be seen that the models are similar, except that the graphical model for the MM model (Figure 4.1) is missing the plate that encompasses the topic proportion, topic label and word variables in the LDA model shown in Figure 2.7. Consequently, in the MM model the topic proportions ($\vec{\alpha}$) are chosen for the entire corpus rather than for each document ($\vec{\theta}_d$) as in the LDA model.

6.2 DATASETS

Two long texts will be use in the experiments. The first is a collection of 13 500 documents from the Associated Press (AP) newswire. These article are about a variety of topics, including politics, economics and medical research. The second corpus is a collection of 495 *Finweek* [1] news articles (De Waal *et al.*, 2007). *Finweek* is a financial magazine and the articles in the corpus are about

[1] http://www.fin24.co.za

companies and markets.

As with the long text, two short texts will be used. Both datasets are collections of tweets. The first dataset is a collection of 4 738 tweets from two users, one against and the other for the controversial *low carb, high fat* diet. The dataset name will be abbreviated as *lchf*. The second corpus is a collection of 77 946 tweets about the weather made available by the *CrowdFlower Open Data Library* (`http://www.crowdflower.com/data-for-everyone`).

6.3 DATA PREPARATION

Before the corpora can be used, they first undergo some preprocessing. This involves the removal of stop words, special characters and numbers. Stop words are words such as *of*, *a* and *but* that do not provide any useful information that can be used in topic assignments. In the same light, words that occur commonly throughout the corpus are also added to the list of stop words. For instance, if the documents in a corpus are articles about lions, then *lions* can be included in the list of stop words as it will arise several times in all the documents thus providing no useful information for the purposes of topic modelling. The special characters that are removed include characters like hash tags, colons, exclamation marks and question marks.

As is typically done in practice, words occurring less than two times in the entire corpus are also removed. Since the vocabulary is very large, many words will have probabilities of occurring in a topic that are close to zero and the $document \times word$ matrix will contain many zeros. This is undesirable as it makes the model unstable, thus removing words with low occurrences helps to improve the performance of the model.

For the preprocessing, we use the *vocabulary.py Python* code available under the MIT license at `https://github.com/karpathy/nipspreview/blob/master/vocabulary.py`. This code also performs lemmatization. This is when different forms (inflections) of a word are grouped as one form of the word. For example, instead of taking the inflections *runs*, *ran* and *running* as individual terms, they are all replaced by their base word, *run*. Lemmatization also helps reduce the sparsity of the $document \times word$ matrix, which increases the stability and performance of the model.

6.4 EXPERIMENTAL SETUP

Table 6.2 shows a summary of the experiments that were performed as well as the respective objective.

Table 6.2: Summary of experiments

Objective	Parameters	Datasets	Evaluation metric
To investigate the convergence properties of the LDA and DMM models on long and short text	$\lambda_\alpha = 0.1$ $\lambda_\beta = 0.1$ $Topics = 50$ $Iterations = 50$ $Repetitions = 10$	AP *Finweek* lchf weather	perplexity
To compare the performance of LDA and DMM models on long and short text	$\lambda_\alpha = 0.1$ $\lambda_\beta = 0.1$ $Topics = 10, 20, ..., 100$ $Iterations = 15$ $Repetitions = 10$	AP *Finweek* lchf weather	coherence entropy
To investigate the effect of λ_α on the LDA and DMM models on long and short text	$\lambda_\alpha = 0, 0.1, 0.2, ..., 1$ $\lambda_\beta = 0.1$ $Topics = 50$ $Iterations = 15$ $Repetitions = 10$	*Finweek* lchf	coherence entropy
To investigate the effect of λ_β on the LDA and DMM models on long and short text	$\lambda_\alpha = 0.1$ $\lambda_\beta = 0, 0.1, 0.2, ..., 1$ $Topics = 50$ $Iterations = 15$ $Repetitions = 10$	*Finweek* lchf	coherence entropy

Both the LDA and MM models are estimated using the Gibbs sampler. For the LDA we

implement the *Python* code, *lda.py*, which is available at `https://github.com/shuyo/iir/blob/master/lda/lda.py` under the MIT license. Since the Gibbs sampler is used for the MM model, it will be referred to as the Dirichlet Multinomial Mixture (DMM) model (Yin and Wang, 2014). The code that we implemented for this is shown in Appendix C1. All *Python* codes are run in *Python* version 2.7.

When applying topic models, an assumption of the number of topics contained in the corpus must be made. This is specified in the parameters column as *Topics*. The number of times that the Gibbs sampler is allowed to run is specified by the *Iterations* parameter and the number of times that the entire experiment is repeated is indicated by the parameter *Repetitions*. Lastly, λ_α and λ_β denote the Dirichlet parameters.

The performance measure that will be used at each stage is shown in the far right hand column. Note, in all our calculations of the coherence and entropy we will use the UMASS metric with $\epsilon = 1$ as was done in the original article by Mimno *et al.* (2011).

6.5 RESULTS

6.5.1 INVESTIGATING THE CONVERGENCE PROPERTIES OF THE LDA AND DMM MODELS ON LONG AND SHORT TEXT

The objective of these experiment is to investigate the convergence properties of the two models. The smoothing parameters λ_α and λ_β are both fixed at 0.1 as in Yin and Wang (2014). The number of topics was also fixed to 50. The choice to use 50 was a subjective one. It is often not favourable to have too many or too few topics, thus 50 seemed to be an appropriate choice. The Gibbs sampler was then run for 50 iterations and the perplexity was calculated each time. This process was then repeated 10 times on each dataset and the perplexities were plotted.

6.5.1.1 *LONG TEXT*

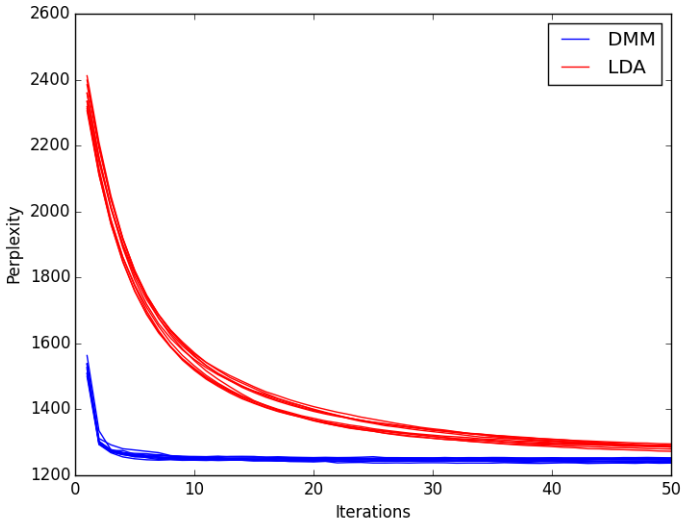


Figure 6.1: Perplexities on AP corpus

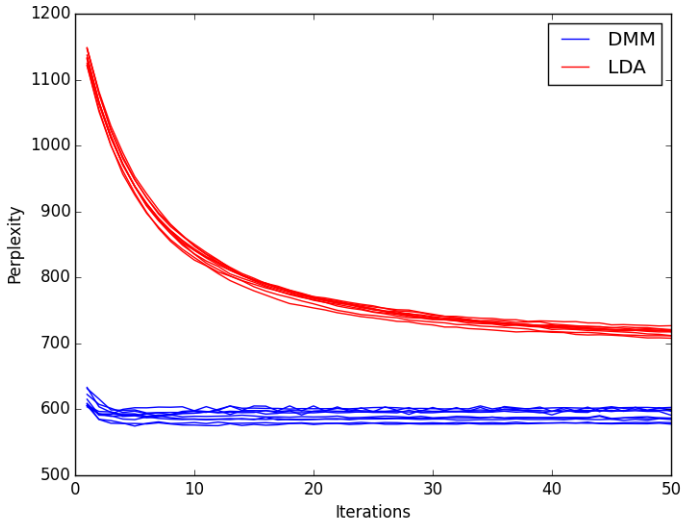


Figure 6.2: Perplexities on *Finweek* corpus

From Figures 6.1 and 6.2, it can be seen that both the models converge as the number of iterations increase. The DMM model converges much faster than the LDA model in both cases. As men-

tioned before, Yin and Wang (2014) stated that the Gibbs sampler is very efficient and can achieve good and stable performance in just 5 iterations. The results appear to support their claim, as it can be seen that the DMM model converges in approximately 5 iterations.

6.5.1.2 SHORT TEXT

A look at the graphs for the short text in Figures 6.3 and 6.4 shows that both models converge, and once again, the DMM does so faster than the LDA model.

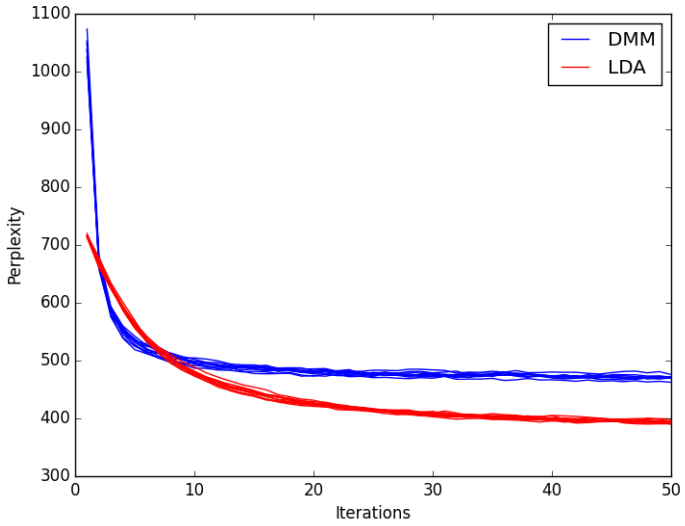


Figure 6.3: Perplexities on lchf corpus

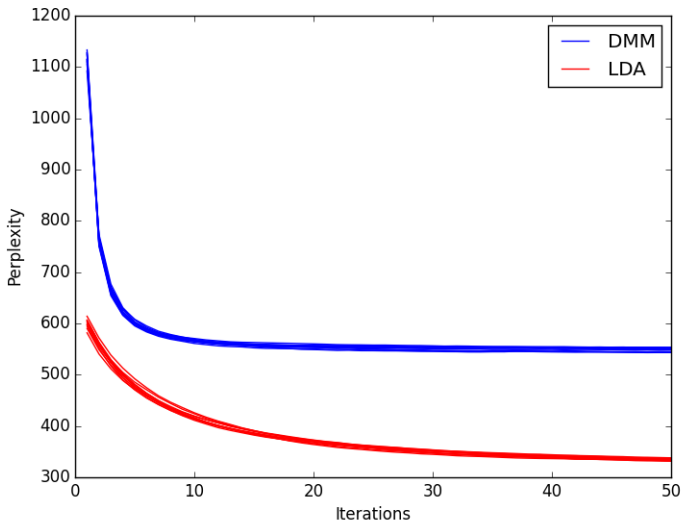


Figure 6.4: Perplexities on weather corpus

6.5.2 COMPARING THE PERFORMANCE OF THE LDA AND DMM MODELS ON LONG AND SHORT TEXT

In the following experiments, we compare the performance of both models using coherence and entropy. The number of topics are varied, $Topics = 10, 20, 30, 40, 50, 100$, whilst the smoothing parameters λ_α and λ_β are both fixed at 0.1. The Gibbs sampler was run for 15 iterations and this process was then repeated 10 times on each dataset, calculating the coherence and entropy each time.

6.5.2.1 *LONG TEXT*

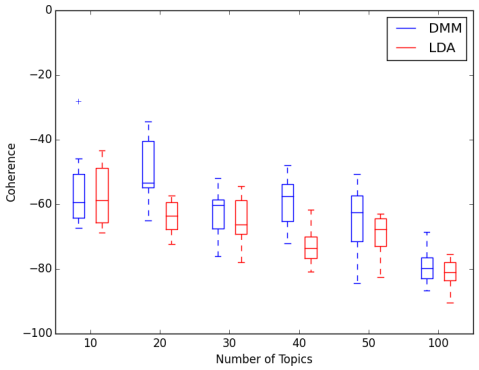


Figure 6.5: Coherence for the AP corpus

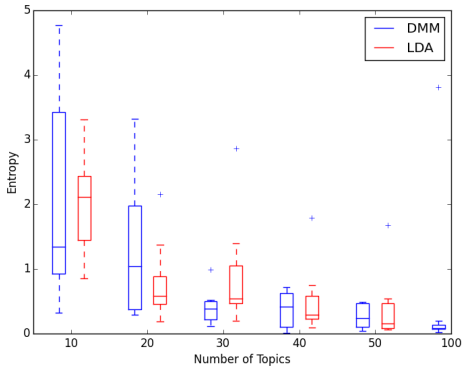


Figure 6.6: Entropy for the AP corpus

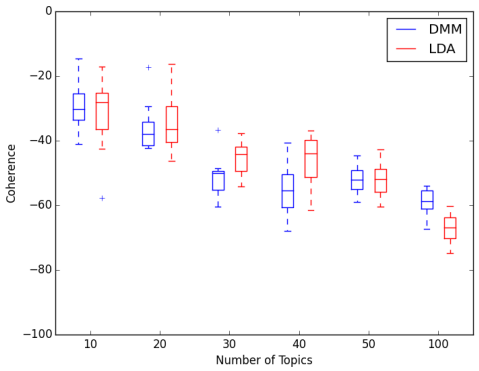


Figure 6.7: Coherence for the *Finweek* corpus

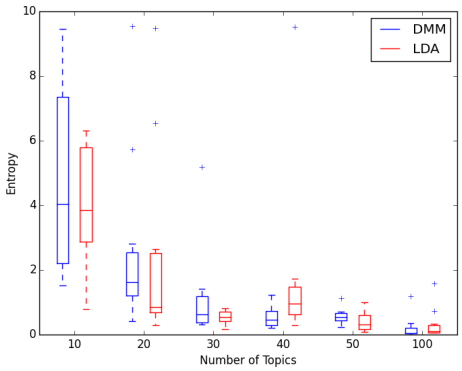


Figure 6.8: Entropy for the *Finweek* corpus

Figure 6.5 shows the coherence values for the models on the AP corpus as the number of topics is varied. It can be seen that the overall coherence of the model decreases for both the LDA and DMM as the number of topics increases, which indicates a decline in the performance of both models as the number of topics increases. It can also be seen that the performance of the LDA and DMM is very similar, with the DMM slightly outperforming the LDA for $Topics = 20, 40, 50$.

From Figure 6.6, it can be seen that the entropies for both models also decrease as the number of topics increases, indicating that the difference between the low and high quality topics being learned gets narrower as more topics are learned. Except for $Topics = 30$, the difference in the

quality of the topics learned by both models is very similar. For $Topics = 30$, the difference in topic quality of the DMM is smaller than that of the LDA. When comparing these results with those presented in Stevens *et al.* (2012) one would notice that the entropies plotted here decrease rather than increase as in Stevens *et al.* (2012). This difference is possibly due to the difference in datasets used. In addition, it can also be seen that the variation in the entropies of the DMM model are quite high for the low topic numbers and it decreases as the number of topics increase. This large variation in entropy implies that the difference in topic quality for small topic numbers can vary largely from experiment to experiment.

In Figures 6.7 and 6.8, we see a similar pattern. Both the coherence and entropy for the *Finweek* corpus drop as the number of topics increases, but we see that the LDA's performance, based on coherence, is generally slightly better than that of the DMM. Only for 100 topics does the DMM outperform the LDA. Once again, the entropies decrease as the number of topics increase and there is no significant difference in the entropies for either model. As with the AP corpus, the variation in the entropies decreases as the number of topics increases, with the DMM generally having greater variation, especially at low topic numbers.

6.5.2.2 SHORT TEXT

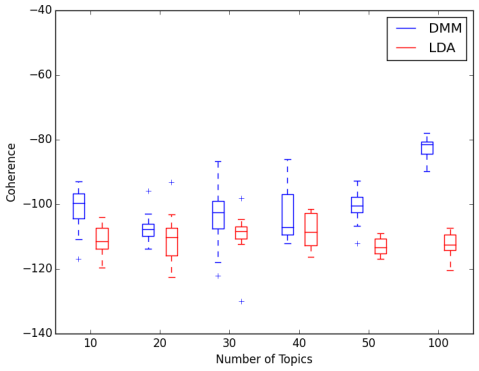


Figure 6.9: Coherence for the lchf corpus

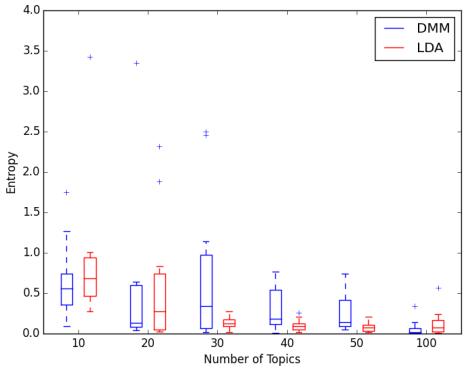


Figure 6.10: Entropy for the lchf corpus

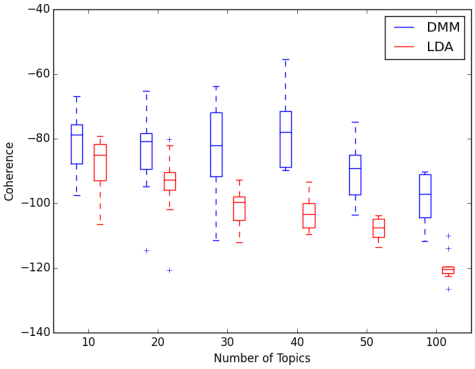


Figure 6.11: Coherence for the weather corpus

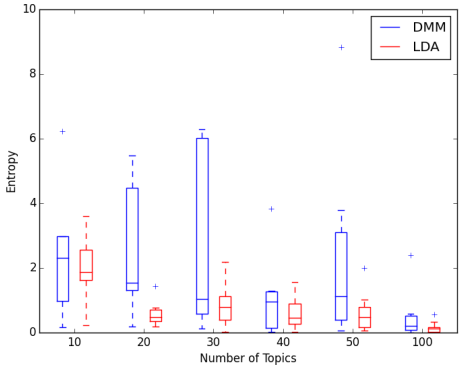


Figure 6.12: Entropy for the weather corpus

The coherences for the short texts show positive results. Looking at Figure 6.9, we see that the DMM generally outperforms the LDA in a majority of cases. In fact, the coherence for the DMM increases as the number of topic increase.

The entropies shown in Figure 6.10 do not appear to fluctuate drastically, but they do decrease gradually indicating that the difference in quality of the topics being learned increases gradually as the number of topics increase. The difference in topic quality remains relatively similar for the LDA as the number of topics increases whereas this difference increases for the DMM.

The results on the weather corpus are similar to those of the other short text corpus. On the weather corpus, the performance of the DMM is mostly better than that of the LDA, as can be seen by the higher coherences of DMM than the LDA in Figure 6.11.

The variation in entropy, as seen in Figure 6.12 is greater for the DMM than for the LDA, and as with the lchf corpus, the entropy also decreases as the number of topics increases. In addition, we also notice that the entropies for the DMM are generally higher than those of the LDA.

6.5.3 INVESTIGATING THE EFFECT OF λ_α ON THE COHERENCE AND ENTROPY OF THE LDA AND DMM MODELS ON LONG AND SHORT TEXT

The aim of this section and the next is to investigate the influence of the Dirichlet parameters, λ_α and λ_β. The number of topics is fixed to 50, the iterations to 15 and λ_β to 0.1. The experiment is then repeated 10 times for each $\lambda_\alpha = 0, 0.1, 0.2, ..., 1$ on one of the long texts, the *Finweek* corpus, and on one of the shorts texts, the weather corpus.

6.5.3.1 LONG TEXT

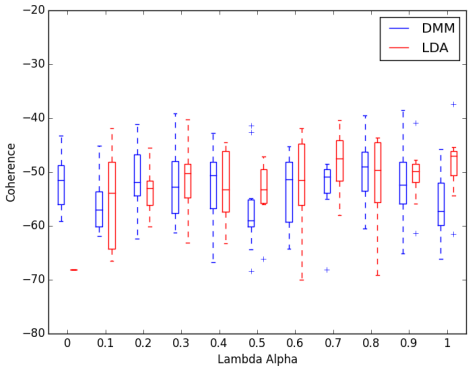


Figure 6.13: Effect of λ_α on the coherence of the *Finweek* corpus

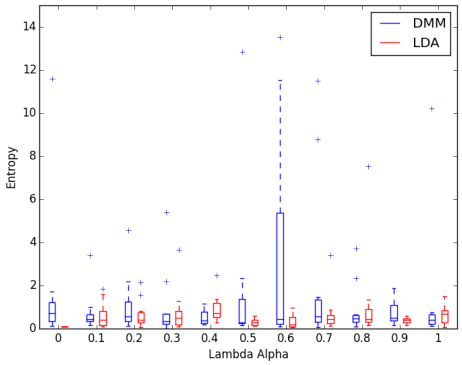


Figure 6.14: Effect of λ_α on the entropy of the *Finweek* corpus

In Figure 6.13, the coherences for both the LDA and DMM appear to increase and drop for $0 < \lambda_\alpha \leq 0.5$ and again for $0.5 \leq \lambda_\alpha < 0.9$ (whilst keeping all other variables fixed). The coherences are more or less the same for all values of λ_α except at $\lambda_\alpha = 0$ where the DMM performs better and at $\lambda_\alpha = 1$ where the LDA performs better. In general, it appears that the performance of both models is very similar.

The entropies for the *Finweek* corpus shown in Figure 6.14 are more or less the same, except at $\lambda_\alpha = 0.6$. On the long text, we can deduce that difference in topic qualities is relatively similar for all values of λ_α for the most part.

6.5.3.2 *SHORT TEXT*

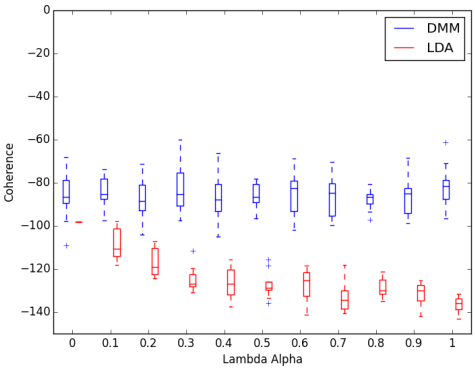


Figure 6.15: Effect of λ_α on the coherence of the weather corpus

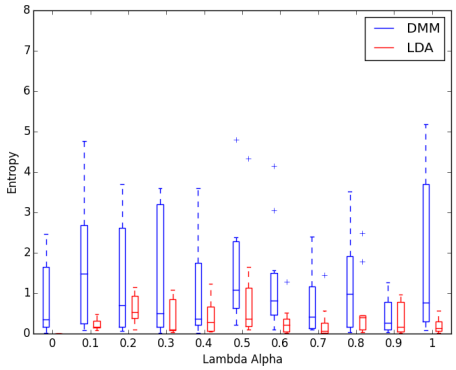


Figure 6.16: Effect of λ_α on the entropy of the weather corpus

On the short text (weather corpus), it can be seen from Figure 6.15 that the DMM has better coherence for all values of λ_α. This implies that the DMM model outperforms the LDA for all values of λ_α. The coherence for the DMM remains fairly constant, whilst that of the LDA decreases as λ_α increases. This implies that the performance of the DMM is relatively unaffected by the choice of λ_α, whereas the performance of the LDA declines as λ_α increases.

The entropies for the DMM are generally higher than those of the LDA, but once again, in Figure 6.16 we observe a greater variation in the entropies for the DMM model than for the LDA. This means that the difference between the low and high quality topics will vary quite a bit for the DMM than for the LDA from repetition to repetition, even if the parameters are fixed.

6.5.4 INVESTIGATING THE EFFECT OF λ_β ON THE COHERENCE AND ENTROPY OF THE LDA AND DMM MODELS ON LONG AND SHORT TEXT

As with the experiments on λ_α, the procedure is the same, except that λ_α is fixed at 0.1 whilst λ_β varies.

6.5.4.1 LONG TEXT

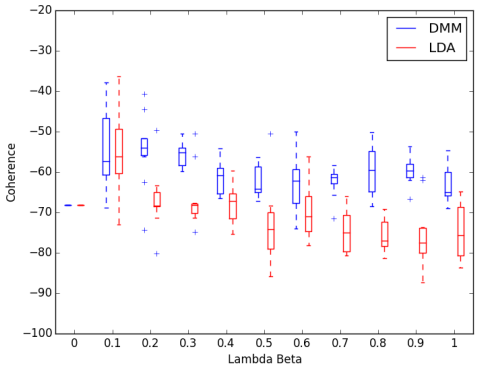


Figure 6.17: Effect of λ_β on the coherence of the *Finweek* corpus

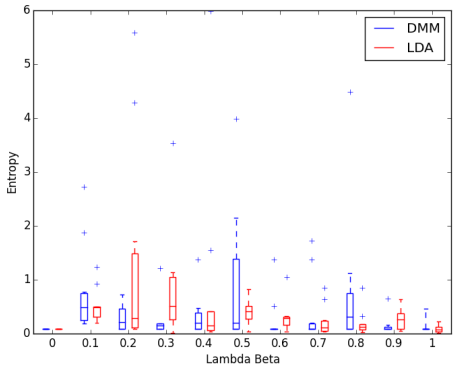


Figure 6.18: Effect of λ_β on the entropy of the *Finweek* corpus

Figure 6.17 shows how the coherence varies as λ_β varies (keeping all other variables fixed). For the long text, the DMM performs better than the LDA for $0.2 \leq \lambda_\beta \leq 1$. The performance is similar for $\lambda_\beta = 0$ and $\lambda_\beta = 0.1$.

In Figure 6.18, it can be seen that the LDA entropies display greater variation than the DMM for $\lambda_\beta = 0.2$ and 0.3, whilst the DMM shows greater variation for $\lambda_\beta = 0.5$ and 0.8. Other than this, there is no clear pattern or trend in the entropies as λ_β varies.

6.5.4.2 *SHORT TEXT*

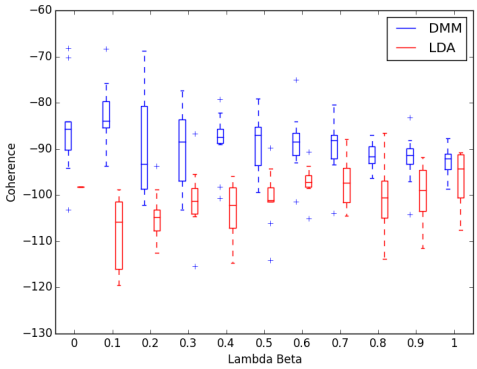


Figure 6.19: Effect of λ_β on the coherence of the weather corpus

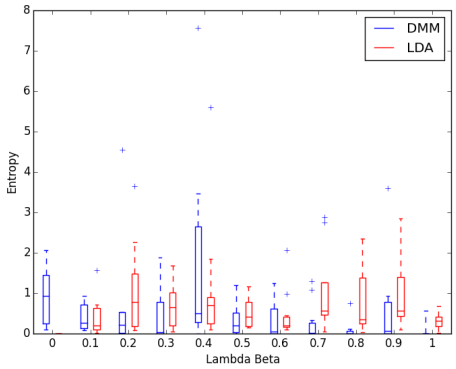


Figure 6.20: Effect of λ_β on the entropy of the weather corpus

On the short text, we see that the coherences of the DMM model are greater than those of the LDA for all values of λ_β in Figure (6.19), which implies that the DMM performs better than the LDA for all values of λ_β. It can also be seen that the coherences for the DMM gradually decrease whilst those of the LDA gradually increase as λ_β increases. Lastly, the entropies shown in Figure 6.20 appear to be slightly higher for the LDA for larger values of λ_β, however for other values of λ_β difference in entropies does no appear significant.

6.5.5 EXAMPLE TOPICS

The tables that follow show 5 out of 50 topics that were inferred by each model. Each topic is represented by the top 10 most probable words. As previously mentioned, the topic names must be assigned manually. This assignment of topic names is a subjective process so the interpretation of the topic described by a group of words may vary from person to person.

Note, although the topics are numbered from 1 to 5, due to the unsupervised nature of topic models they do not necessarily correspond to the same topic numbers in different models. For example, Topic 1 for the LDA is not necessarily the same topic described in Topic 1 of the DMM model. The topics are labelled in this manner for ease of reference.

In implementing the models, each model was run once with the number of topics fixed at 50, λ_α and λ_β both fixed at 0.1 for 15 iterations.

6.5.5.1 *AP CORPUS*

Table 6.3: Top 10 words from some of the inferred topics for the AP corpus

LDA				
Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
space	mr	police	service	bush
shuttle	hospital	year	department	president
mission	life	car	law	house
launch	doctor	home	office	senate
nasa	state	people	justice	reagan
time	death	killed	federal	budget
year	mecham	family	time	defense
day	family	death	year	administration
mile	victim	man	public	white
earth	told	back	business	congress
DMM				
Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
aid	venus	cent	percent	black
percent	magellan	future	year	people
drug	scientist	lower	million	government
health	spacecraft	oil	billion	year
disease	earth	higher	rate	white
year	picture	price	price	mandela
people	test	market	month	south
virus	time	soybean	sale	court
researcher	nosair	bushel	increase	group
blood	year	contract	quarter	police

For the LDA, Topic 1 appears to refer to a mission to outer space whilst Topic 3 appears to refer to a crime. The DMM appears to have also caught a topic to do with outer space travel, Topic 2, as well as a topic that appears to refer to health research.

6.5.5.2 *FINWEEK CORPUS*

Table 6.4: Top 10 words from some of the inferred topics for the *Finweek* corpus

LDA				
Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
uranium	year	gold	cash	people
profit	earnings	anglogold	flow	business
sale	time	field	sheet	employee
investment	cash	deep	asset	strategy
banking	result	ashanti	balance	transformation
higher	growth	oz	operating	south
income	price	u	profit	education
financial	dividend	stake	interim	gib
volume	financial	south	statement	knowledge
strong	performance	barrick	line	economy
DMM				
Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
director	bank	telkom	mining	people
share	market	business	company	growth
shareholder	banking	share	gold	education
dorbyl	share	deal	listing	research
year	year	connexion	simmer	economic
company	business	year	year	time
executive	group	company	jse	culture
remgro	growth	market	firm	human
time	sa	shareholder	empowerment	gdp
management	rate	term	asset	world

Topic 2 of the LDA seems to be about company financial status whilst Topic 5 could possibly refer to the economy. For the DMM, Topics 2 and 5 appear to be about banking and the economy, respectively.

6.5.5.3 *LCHF CORPUS*

Table 6.5: Top 10 words from some of the inferred topics for the lchf corpus

LDA				
Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
sugar	real	free	lchf	science
amp	meal	lchf	share	prize
heart	food	grain	lowcarb	nobel
disease	revolution	sugar	realfood	winner
life	carbs	chocolate	carb	physic
key	fat	amp	low	problem
markhymanmd	eat	share	lunch	story
annchildersmd	people	sugarfree	wheatfree	michael
fed	obesity	wheat	idea	higgs
lead	choice	banting	easy	dietary
DMM				
Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
lchf	nobel	insulin	evidence	fat
realfood	prize	resistance	science	carbs
share	sa	ir	noakes	diet
chocolate	year	high	clinical	food
baked	scientist	diabetes	trial	eat
cheesecake	human	carbs	study	protein
fabulous	physic	carb	diet	day
time	amp	disease	georgeclaassen	eating
lowcarb	higgs	diet	helpdietsa	low
wheatfree	michael	glucose	research	carb

From Table 6.5, Topic 5 in the LDA and Topic 2 of the DMM appear to be about the Nobel Prize winner in Physics, Michael Higgs. Topic 5 for the DMM may be about diet whilst Topic 2 for the LDA could possibly refer to eating habits.

6.5.5.4 *WEATHER CORPUS*

Table 6.6: Top 10 words from some of the inferred topics for the weather corpus

LDA				
Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
mention	sunny	storm	day	weather
weather	degree	tornado	sunny	link
lt	link	mention	rainy	week
sunshine	storm	joplin	beautiful	state
great	chance	moving	good	news
sunny	forecast	lightning	today	wisconsin
love	cloudy	crazy	work	city
back	sky	dallas	happy	forecast
send	morning	shelter	enjoy	kansa
beautiful	shower	friend	sunday	beach
DMM				
Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
day	mention	high	storm	f
rainy	weather	low	link	mention
mention	sunshine	current	mention	rain
weather	sunny	condition	victim	wind
sunny	link	sunny	alabama	temp
link	storm	weather	damage	mph
cold	snow	forecast	weather	hum
good	today	forecast	tornado	inch
today	rain	fair	shelter	baro
love	day	cloudy	relief	az

As would be expected, the topics inferred for the weather corpus contain lots of words related to the weather. The LDA Topic 1 could refer to beautiful weather conditions whilst Topic 3 refers to bad weather conditions, perhaps in Dallas. Lastly, Topic 5 of the DMM has many words to do with measuring different aspects of the weather such as temperature and humidity, and Topic 4 also seems to refer to bad weather, possibly in Alabama.

Generally, both the LDA and DMM produce some interpretable topics. Unfortunately, we can not compare models by simply looking at the topics. We would need measures such as word

intrusion for that purpose or a labelled dataset to compare the inferred results with the true labels.

6.6 SUMMARY OF RESULTS

What follows is summary of the overall results of our experiments.

- Based on perplexity, both the DMM and MM models converge when applied to both long and short text corpora.
- Based on coherence, on the long text, the performance of the DMM and LDA was very similar, with the LDA performing slightly better. On the other hand, the DMM performed much better than the LDA on the short texts.
- The entropies for the DMM generally display greater variation than those of the LDA on both long and short text. Thus, the difference in quality of the low and high quality topics will be inconsistent and will vary more for the DMM from experiment to experiment.
- The entropies for both models are similar on the long texts. Although, the entropies for the DMM are somewhat higher on the short texts than for the LDA. Furthermore, the DMM entropies generally displayed larger variation, which is not a favourable characteristic.
- For the long text, the entropies for both the LDA and DMM display a downward trend as the number of topics increase. The entropies for both models display a more random pattern on the short text, which would imply that the difference in quality of topics may be less predictable from the number topics.
- Based on coherence, the performance of the LDA and DMM on the long text was similar for most values of λ_α, but the LDA was outperformed by the DMM on the short text for all values of λ_α. On the long text, the entropies were similar for both models for most values of λ_α. On the other hand, the entropies for the DMM on the short texts were higher than those from the LDA for all λ_α and the variation in entropies for the DMM was large.
- Based on coherence, for both the long text and short text, the performance of the DMM was better than that of the LDA for all λ_β. On both corpora, the influence of λ_β on the entropy did not display any clear pattern.

- Both models produce some interpretable topics, but there is a need to formally assess them, which we intend to do as part of our future work.

6.7 CONCLUSION

From our experiments the results are positive. Based on coherence the LDA performs better than the DMM on the long text. For the short text, the MM model is seen to perform better than the LDA in most cases, based on both the coherence.

From our experiments on the influence of the Dirichlet parameters, we found that the choice of λ_α does not greatly affect the coherence of the DMM model on long or short text. For the long text, the coherence values of the LDA are also unaffected by λ_α, yet for the short text, caution would be advised as the performance of the LDA decreases as λ_α increases. With regards to entropy, changing λ_α does not affect either model much for both long and short text.

On the long text, the coherence of both models decreases as λ_β increases, which would indicate the need to avoid choosing λ_β too large. On the short text, the coherence for the DMM still decreases yet that of the LDA increases, which suggests that it may be favourable to use higher λ_β's for the LDA when applying it to short text. Looking at entropy, the pattern for both models on both texts appears random, which does not point to a clear guideline as to how to choose λ_β to obtain good entropies. Further investigation would be needed here.

CHAPTER SEVEN

CONCLUSIONS

7.1 INTRODUCTION

As was mentioned in Chapter 1, the aim of this research was to answer the following question:

> *Given the assumption of each document belonging to at most one topic, can the hidden themes in a short text corpus be identified better by the Multinomial Mixture model than by its more sophisticated counterparts in a practical setting?*

To address this question we applied both the Latent Dirichlet Allocation model and the Multinomial Mixture model on two long text and two short text corpora. Both models were estimated using the Gibbs sampler and, based on perplexity, which is a function of the log-likelihood of the respective model, both algorithms were found to converge quite quickly with the LDA taking more iterations than the MM (also referred to as the DMM, Dirichlet Multinomial Mixture model, when estimated using the Gibbs sampler).

We then compared the performance of the two models based on the coherence and entropy. In summary, the performance of the LDA was only slightly better than that of the DMM on the long texts. However, on the short text, the performance of the DMM was distinctly better than that of the LDA. Based on entropy on the long text, the difference between the quality of the topics

learned does not appear to differ significantly for the two models. On the short text, the difference in topic qualities for the DMM is slightly larger than that of the LDA. As was mentioned earlier, whether large or small differences in topic quality are desired or not is based on user preference. It was also observed that the variation in the DMM entropies is usually large, which is generally not a favourable characteristic. Furthermore, it was found that varying λ_α does not affect the coherence and entropy significantly when applying the DMM to short text thus suggesting that any $0 \leq \lambda_\alpha \leq 1$ may be acceptable. The effect of λ_β on the entropy is not so clear and requires further investigation, however, the results indicate that choosing smaller values of $\lambda_\beta > 0$ will give good coherence for the DMM on short text. (The reason for not allowing $\lambda_\beta = 0$ is explained under the *Interpretation of the Dirichlet parameters* section on page 41.)

As much as these results appear to point in the direction of the Multinomial Mixture being a better model for short text than the LDA, due to the unsupervised nature of the topic models, their evaluation still remains a major challenge. There does not yet exist a widely accepted set of standard evaluation methods for all the different aspects of topic models, such as information retrieval and generalization. Nonetheless, at this point in time, based on our tests, the results of our research were positive.

In response to the aforementioned research question, it appears that the Multinomial Mixture model is a viable model for handling short text corpora for the purposes of understanding its thematic structure. Based on our experiments, when comparing coherences we find the MM to be a better model than the LDA for short text. Nonetheless, there is still a need to consider other evaluation methods to fully assess all the aspects of the MM model, such as its classification and generalization capabilities compared to those of the LDA.

7.2 CONTRIBUTIONS

The amount of available literature that addresses topic modelling on short text is relatively small, however, as short text continues to increase in relevance, the amount of literature is rapidly growing. In view of this research gap, the fact that this work addresses an aspect of the short text problem is the main contribution of this work.

Secondly, given the generative assumption of the MM model, that a document can belong to only one topic, based on coherence, the results of our study give an indication of the MM model

being a better model for short text than the LDA, which assumes multiple topics per document.

7.3 FUTURE WORK

As with any research project, this work has made us aware of issues and limitations that will need to be addressed and studied further.

- Although we only used coherence and entropy to evaluate the models, in Chapter 5 we see that many evaluation methods exist each with different capabilities, accordingly, there is a need to compare the LDA and MM using other measures.
- One specific, important area that we did not test was the ability of the MM model to generalize on unseen text compared to the LDA. Once again, the unsupervised nature of topic models imposes the challenge of this being a difficult aspect to evaluate. Researchers such as, Wallach *et al.* (2009), have proposed measures for this purpose, however, whether it is possible to effectively evaluate the generalization capabilities of a topic model is an area of debate. Consequently, this is an area that we would like to investigate further in future.
- Another limitation of our study is the fact that we only compare the MM to the LDA. In view of this, it would be of value to compare the performance of the MM with other models, such as Probabilistic Latent Semantic Analysis and the Gamma-Poisson.
- We would also like to explore ways of improving on our implementation of the Dirichlet Multinomial Mixture model to enable it to handle large corpora faster. Possible methods include parallel computing and having a computer programmer optimize the code. In addition, following a similar approach to the online LDA algorithm proposed by Hoffman *et al.* (2010), one could possibly attempt to create an online version of the MM model.
- In our study, we unfortunately did not have any labelled corpora on which to test our hypothesis, as we did not have access to a suitable labelled corpus. This creates another area of future work. We would like to compare the results of the MM and LDA with the true document labels.
- Lastly, it would also be of interest to take this work a step further by exploring the use of the MM model in other applications, such as sentiment analysis.

7.4 CONCLUSION

As short text continues to become increasingly pertinent, the need for tools that can effectively handle short text is becoming increasingly important. In order to address the short text problem in text analytics as whole, it is necessary to gain a full understanding of the difficulties that short text poses. Through this work, we have gained a bit more insight into the challenges that entail the application of topic models on short text and have reason to propose the application of the Multinomial Mixture model as a way of tackling some of them.

REFERENCES

Bermingham, A. and Smeaton, A.F. (2011). On using twitter to monitor political sentiment and predict election results.

Bishop, C.M. *et al.* (2006). *Pattern recognition and machine learning*, vol. 1. springer New York.

Blei, D.M. (2004). *Probabilistic models of text and images*. Ph.D. thesis, University of California.

Blei, D.M. (2012). Probabilistic topic models. *Communications of the ACM*, vol. 55, no. 4, pp. 77–84.

Blei, D.M., Ng, A.Y. and Jordan, M.I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, vol. 3, pp. 993–1022.

Bollen, J., Mao, H. and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8.

Canny, J. (2004). Gap: a factor model for discrete data. In: *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 122–129. ACM.

Casella, G. and George, E.I. (1992). Explaining the gibbs sampler. *The American Statistician*, vol. 46, no. 3, pp. 167–174.

Castella, Q. and Sutton, C. (2014). Word storms: Multiples of word clouds for visual comparison of documents. In: *Proceedings of the 23rd international conference on World wide web*, pp. 665–676. International World Wide Web Conferences Steering Committee.

Chang, J., Gerrish, S., Wang, C., Boyd-graber, J.L. and Blei, D.M. (2009). Reading tea leaves: How humans interpret topic models. In: *Advances in Neural Information Processing Systems*, pp. 288–296.

Chen, H., Chen, J. and Kalbfleisch, J.D. (2004). Testing for a finite mixture model with two components. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 66, no. 1, pp. 95–115.

Cui, W., Wu, Y., Liu, S., Wei, F., Zhou, M.X. and Qu, H. (2010). Context preserving dynamic word cloud visualization. In: *Pacific Visualization Symposium (PacificVis), 2010 IEEE*, pp. 121–128. IEEE.

Darling, W.M. (2011). A theoretical and practical implementation tutorial on topic modeling and gibbs sampling. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 642–647.

De Waal, A. (2010). *Topic Models with Structured Features*. Ph.D. thesis, North-West University, Potchefstroom Campus.

De Waal, A., Barnard, E. and du Preez, E. (2007). Topic models applied to multilingual data. In: *Proceedings of the 18h Annual Symposium of the Pattern Recognition Association of South Africa (PRASA)*, pp. 99–103.

Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W. and Harshman, R.A. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, , no. 6, pp. 721–741.

Gerber, M.S. (2014). Predicting crime using twitter and kernel density estimation. *Decision Support Systems*, vol. 61, pp. 115–125.

Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J. and Tibshirani, R. (2009). *The Elements of Statistical Learning*, vol. 2. Springer.

Havre, S., Hetzler, B. and Nowell, L. (2000). Themeriver: Visualizing theme changes over time. In: *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pp. 115–123. IEEE.

Heinrich, G. (2005). Parameter estimation for text analysis. Tech. Rep., Technical report.

Hoffman, M., Bach, F.R. and Blei, D.M. (2010). Online learning for latent dirichlet allocation. In: *Advances in Neural Information Processing Systems*, pp. 856–864.

Hofmann, T. (1999*a*). Probabilistic latent semantic analysis. In: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 289–296. Morgan Kaufmann Publishers Inc.

Hofmann, T. (1999*b*). Probabilistic latent semantic indexing. In: *Proceedings of the 22nd Annual International ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57. ACM.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, vol. 42, no. 1-2, pp. 177–196.

Hong, L. and Davison, B.D. (2010). Empirical study of topic modeling in twitter. In: *Proceedings of the First Workshop on Social Media Analytics*, pp. 80–88. ACM.

Huang, A., Milne, D., Frank, E. and Witten, I.H. (2009). Clustering documents using a wikipedia-based concept representation. In: *Advances in Knowledge Discovery and Data Mining*, pp. 628–636. Springer.

Jun, H.Y., Xin, J.J. and You, C.H. (2013). Chinese short-text classification based on topic model with high-frequency feature expansion. *Journal of Multimedia*, vol. 8, no. 4, pp. 425–431.

Kamper, H. (2013). Gibbs sampling for fitting finite and infinite gaussian mixture models.

Khartabil, D. (2013). Data mining and visualisation of twitter using topic modelling.

Li, M. and Zhang, L. (2008). Multinomial mixture model with feature selection for text clustering. *Knowledge-Based Systems*, vol. 21, no. 7, pp. 704–708.

Lin, C. and He, Y. (2009). Joint sentiment/topic model for sentiment analysis. In: *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 375–384. ACM.

Lindsay, B.G. (1995). Mixture models: theory, geometry and applications. In: *NSF-CBMS regional conference series in probability and statistics*, pp. i–163. JSTOR.

Mehrotra, R., Sanner, S., Buntine, W. and Xie, L. (2013). Improving lda topic models for microblogs via tweet pooling and automatic labeling. In: *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pp. 889–892. ACM.

Meilă, M. (2003). Comparing clusterings by the variation of information. In: *Learning theory and kernel machines*, pp. 173–187. Springer.

Mimno, D., Wallach, H.M., Talley, E., Leenders, M. and McCallum, A. (2011). Optimizing semantic coherence in topic models. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 262–272. Association for Computational Linguistics.

Newman, D., Chemudugunta, C., Smyth, P. and Steyvers, M. (2006). Analyzing entities and topics in news articles using statistical topic models. In: *Intelligence and Security Informatics*, pp. 93–104. Springer.

Newman, D., Noh, Y., Talley, E., Karimi, S. and Baldwin, T. (2010). Evaluating topic models for digital libraries. In: *Proceedings of the 10th annual joint conference on Digital libraries*, pp. 215–224. ACM.

Nigam, K., McCallum, A.K., Thrun, S. and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine learning*, vol. 39, no. 2-3, pp. 103–134.

Pennisi, E. (1996). Seeking life's bare (genetic) necessities. *Science*, vol. 272, no. 5265, pp. 1098–1099.

Rigouste, L., Cappé, O. and Yvon, F. (2007). Inference and evaluation of the multinomial mixture model for text clustering. *Information processing & management*, vol. 43, no. 5, pp. 1260–1280.

Stevens, K., Kegelmeyer, P., Andrzejewski, D. and Buttler, D. (2012). Exploring topic coherence over many models and many topics. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 952–961. Association for Computational Linguistics.

Wallach, H.M., Murray, I., Salakhutdinov, R. and Mimno, D. (2009). Evaluation methods for topic models. In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1105–1112. ACM.

Xiang, B. and Zhou, L. (2014). Improving twitter sentiment analysis with topic-based mixture modeling and semi-supervised training. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pp. 434–439.

Yin, J. and Wang, J. (2014). A dirichlet multinomial mixture model-based approach for short text clustering. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 233–242. ACM.

Yu, X. (2009). Derivation of gibbs sampling for finite gaussian mixture model.

APPENDIX A

A1. The SAS code used to generate the observations in Example 1:

```
proc iml;
n=1000; p1=0.2; p2=0.3;
p3=1-p1-p2;
y1=3+normal(j(n*p1,1,1));
y2=1.5+normal(j(n*p2,1,1))*sqrt(1);
y3=-2+normal(j(n*p3,1,1))*sqrt(0.2);
y=y1//y2//y3;
create example.y from y;
append from y;
quit;
```

A2. The SAS program to implement the EM algorithm for 3-component Gaussian mixture model is shown below. Note, the indices of the estimated values may not correspond to the indices that were used in defining the random variables, but the results will still be correct. This phenomenon is known as label switching.:

```
proc iml ;
use example.y;
read all into y;

n=nrow(y);
ybar=sum(y)/n;
var1=1/(n)*sum((y-ybar)##2);
var2=var1;
var3=var1;

*to select a random observation as the initial values for the y's;
i1=int(n*ranuni(1))+1;
mu1=y[i1];

i2=int(n*ranuni(1))+1;
mu2=y[i2];

i3=int(n*ranuni(1))+1;
```

```
mu3=y[i3];
print ' The initial values used are: ' mu1 mu2 mu3;

*pi1 pi2 pi3 are the mixing probabilities alpha_k k=1,2,3;
pi1=1/3; pi2=1/3; pi3=1/3;

iterations=0;
do until(tolerance1<10**(-20));
iterations=iterations+1;
*E-step;
sum=pi1*pdf('normal',y,mu1,sqrt(var1))
+pi2*pdf('normal',y,mu2,sqrt(var2))
+pi3*pdf('normal',y,mu3,sqrt(var3));
g1=pi1*pdf('normal',y,mu1,sqrt(var1))/ sum;
g2=pi2*pdf('normal',y,mu2,sqrt(var2))/ sum;
g3=pi3*pdf('normal',y,mu3,sqrt(var3))/ sum;
*print g;

*M-step;
mu1=sum(g1#y)/sum(g1);
mu2=sum(g2#y)/sum(g2);
mu3=sum(g3#y)/sum(g3);

var1=sum(g1#(y-mu1)##2)/sum(g1);
var2=sum(g2#(y-mu2)##2)/sum(g2);
var3=sum(g3#(y-mu3)##2)/sum(g3);

pi1_1=pi1;
pi1=sum(g1)/n;
pi2=sum(g2)/n;
pi3=sum(g3)/n;
tolerance1=abs(pi1_1-pi1);
end;

print iterations;
print mu1 mu2 mu3;
print var1 var2 var3;
print pi1 pi2 pi3;

quit;
```

A3. The SAS program to implement the Gibbs sampler for 3-component Gaussian mixture model is shown below. Note, label switching may also occur when implementing the Gibbs sampler.

```
proc iml ;
use example.y;
read all into y;
```

```
*print y;
n=nrow(y);

var1=1;
var2=1;
var3=1;

*initial values;
mu1=0;
mu2=0;
mu3=0;
*print mu1 mu2 mu3;

*pi1 pi2 pi3 are the mixing probabilities alpha_k k=1,2,3;
pi1=1/3; pi2=1/3; pi3= 1/3;

iterations=0;
do it=1 to 10000;
iterations=iterations+1;

*estimating gamma_ik;
sum=pi1*pdf('normal',y,mu1,sqrt(var1))
+pi2*pdf('normal',y,mu2,sqrt(var2))
+pi3*pdf('normal',y,mu3,sqrt(var3)); *print sum;
g1=pi1*pdf('normal',y,mu1,sqrt(var1))/ sum;
g2=pi2*pdf('normal',y,mu2,sqrt(var2))/ sum;
g3=pi3*pdf('normal',y,mu3,sqrt(var3))/ sum;
*print g1 g2 g3;

* latent variables;
free z;
do c=1 to n;
z = z//RANDMULTINOMIAL(1,1,(g1[c]||g2[c]||g3[c]));
end;
*print z;

*estimates for mu_k;
mu1_1=mu1;
mu1=sum(z[,1]#y)/sum(z[,1]);
mu2=sum(z[,2]#y)/sum(z[,2]);
mu3=sum(z[,3]#y)/sum(z[,3]);

alph_1=sum(z[,1])/n;
alph_2=sum(z[,2])/n;
alph_3=sum(z[,3])/n;
tolerance1=abs(mu1_1-mu1);
end;
print iterations;
print mu1 mu2 mu3;
quit;
```

APPENDIX B

B1. The example $document \times word$ matrix on which the log-likelihoods for the EM algorithm and Gibb sampler were calculated in Chapter 4:

$$\begin{bmatrix} 0 & 0 & 3 & 3 & 1 & 1 & 2 & 0 & 4 & 2 \\ 4 & 2 & 0 & 1 & 0 & 1 & 2 & 2 & 1 & 1 \\ 0 & 1 & 0 & 2 & 4 & 2 & 4 & 3 & 0 & 0 \\ 2 & 0 & 4 & 4 & 0 & 2 & 3 & 3 & 1 & 3 \\ 0 & 0 & 2 & 0 & 1 & 4 & 2 & 1 & 4 & 0 \\ 0 & 0 & 0 & 2 & 0 & 3 & 3 & 1 & 0 & 1 \\ 3 & 0 & 3 & 0 & 1 & 3 & 1 & 1 & 2 & 4 \\ 4 & 2 & 3 & 0 & 2 & 4 & 1 & 2 & 1 & 0 \\ 1 & 4 & 1 & 2 & 2 & 3 & 4 & 3 & 4 & 2 \\ 3 & 0 & 4 & 1 & 1 & 2 & 0 & 1 & 4 & 3 \end{bmatrix}$$

B2. The Python code used to implement the EM algorithm on the test matrix (saved as *test_matrix.csv*) and output the log-likelihood: (Note, λ_α and λ_β are simply *alpha* and *beta* in the code.)

```
    """
    Implements multinomial mixture as described in:
"Rigouste, L, Cappe, O and Yvon, F. Inference and
      evaluation of the multinomial mixture model for
      text clustering. Information Processing and Management, 2007"

    """

import numpy as np
from sklearn.preprocessing import normalize

#np.random.seed(100000001)
```

```
class MixMULT:

    def __init__(self, mat, T):
        """
        Arguments:
        T: Number of topics
        vocab: A set of words to recognize. When analyzing
        documents, any word not in this set will be ignored.
        matrix: The document by words matrix created during
        the preprocessing step.
        """
        #initialization
        self._T = T
        self._mat = np.matrix(mat)
        self._W = self._mat.shape[1]
        self._l = self._mat.sum()
        self._D = self._mat.shape[0]
        self._theta = np.random.uniform(0,1,(self._T, self._D))
        self._theta = normalize(self._theta, norm='l1', axis = 0, copy = False)


    def init_m(self):
        #to calculate first alpha and beta:
        (alpha, beta) = self.do_m(self._theta)
        return alpha, beta

    def do_e(self, alpha, beta):

        #e-step
        logt = np.log(alpha)
        multinomial = np.tile(logt.T,(1,self._D))+np.log(beta).T*self._mat.T
        max_multinomial = np.max(multinomial,axis = 0)
        post_theta2=np.exp(multinomial - (max_multinomial + 20))
        post_theta=normalize(post_theta2, norm='l1', axis = 0, copy = False)
        log_likelihood= self.do_log_likelihood(post_theta, multinomial, self._l)

        return post_theta, log_likelihood

    def do_m(self, post_theta):

        alpha = np.sum(post_theta, axis = 1).reshape(1,-1)
        alpha = normalize(alpha, axis = 1,norm='l1', copy = False)

        beta = self._mat.T*post_theta.T+0.2
        beta = normalize(beta, axis = 0, norm='l1', copy = False)

        return alpha, beta

    def do_log_likelihood(self, post_theta, multinomial, l):
```

```
        log_likelihood=(np.multiply(post_theta, multinomial).sum())
        return log_likelihood

def main():

    nTopics = 5
    test_file = "test_matrix.csv"
    iterations = 50

    f=open(test_file, 'r')
    X = [ map(int,line.split(',')) for line in f ]

    # Initialize the mixture of multinomial algorithm
    mm =  MixMULT(X, int(nTopics))

    #calculate initial alpha and beta:
    (alpha, beta) = mm.init_m()

    #Do the EM:
    for i in range(0,iterations):
        #e-step:
        (theta, log_likelihood) = mm.do_e(alpha, beta)
        print "Iteration %d: " % (i), log_likelihood
        #m-step
        (alpha, beta) = mm.do_m(theta)

if __name__ == "__main__":
    main()
```

B3. The *Python* code used to implement the Gibbs sampler on the test matrix (saved as *test_matrix.csv*) and output the log-likelihood: (Note, λ_α and λ_β are simply *alpha* and *beta* in the code.)

```
import numpy as np
from sklearn.preprocessing import normalize

class DMM:

    """
    Implements multinomial mixture as described in:
"Yin, J. and Wang, J. A dirichlet multinomial mixture model-based
         approach for short text clustering, 2014"

    """

    def __init__(self, docs, T, alpha, beta):
        self.T = T #number of topics
        self.alpha = alpha # parameter of topics prior
```

```
        self.beta = beta   # parameter of words prior
        self.docs = np.matrix(docs) # doc x word matrix
        self.D = self.docs.shape[0] #number of documents
        self.V = self.docs.shape[1] #number of words in vocabulary
        self.N_d = self.docs.sum(axis=1) #total number of word occurences
        #in each document
        self.l = self.docs.sum()#total number of word occurrences in corpus

        self.z = np.matrix(np.random.multinomial(1, [1/float(self.T)]*self.T, self.D))
        #topic assignments for each document
        self.m_z = self.z.sum(axis=0) #number of documents in topic z
        self.n_z_w = self.z.T*self.docs #number of occurrences of word w in topic z
        self.n_z = self.n_z_w.sum(axis=1).T #number of words in topic z

    def inference(self):
        self.theta_r = np.zeros([self.D, self.T])
        for d in range(0, self.D):

            z_d=self.z[d,:]
            z_d_t = np.argmax(z_d) #cluster number of doc d

            m_z = self.m_z - z_d #update m_z

            n_z_w = self.n_z_w # update n_z_w
            for w in range(0, self.V):
                n_z_w[z_d_t, w]=n_z_w[z_d_t, w]-self.docs[d, w]

            n_z = n_z_w.sum(axis=1).T #update n_z

            #sample new topic for doc
            probabilities = np.zeros(self.T)

            for z in range(0, self.T):
                num1 = m_z[0, z] + self.alpha
                #print "num1: ", num1

                den1 = self.D - 1 + self.T*self.alpha
                #print "den1: ", den1

                den2 = 1
                for i in range(0, self.N_d[d]):
                    den2 = den2 * (n_z[0, z] + self.V*self.beta + i)
                #print "den2: ", den2

                num2 = 1
                for ww in range(0, self.V):
                    if self.docs[d, ww] != 0 :
                        for j in (0, self.docs[d, ww]):
                            num2 = num2 *(n_z_w[z, ww] + self.beta + j)
```

```
                probabilities[z] = (num1/den1)*(num2/den2)
                #print "num2: ", num2

            #print probabilities
            probabilities = probabilities/np.sum(probabilities)
            #print "probabilities: ", probabilities

            new_z_d=np.eye(self.T, dtype=int)[np.argmax(probabilities),:]

            self.z[d, :] = self.z[d, :] + new_z_d
            self.m_z = self.z.sum(axis=0) #number of documents in topic z
            self.n_z_w = self.n_z_w + self.z.T*self.docs
            #number of occurrences of word w in topic z
            self.n_z = self.n_z_w.sum(axis=1).T #number of words in topic z
            self.theta_r[d, :] = probabilities

    def worddist(self):
        """get topic-word distribution"""
        beta_r = np.zeros([self.T, self.V]) #same as beta from rigouste
        for zz in range(0, self.T):
            for www in range(0, self.V):
                beta_r[zz,www] = (self.n_z_w[zz,www]+self.beta)
                /(self.n_z[0,zz] + self.V*self.beta)
        return beta_r


    def do_log_likelihood(self, post_theta, multinomial, l):
        log_likelihood=( np.multiply(post_theta, multinomial).sum())
        return log_likelihood

    def log_likelihood(self):

        beta = self.worddist()
        alpha_r = (self.n_z + 0.1) / np.sum((self.n_z + 0.1))
        logt = np.log(alpha_r)
        multinomial = np.tile(logt.T,(1,self.D))+np.log(beta)*self.docs.T

        #log_likelihood
        log_likelihood = self.do_log_likelihood(self.theta_r.T, multinomial, self.l)
        return log_likelihood


    def dmm_learning(self, dmm, iteration):
        for i in range(iteration):
            (labels) = self.inference()
            p=self.log_likelihood()
            print "Iteration %d: " % (i), p

def main():
```

```
    iteration = 50
    nTopics = 5
    test_file = "test_matrix.csv"

    f=open(test_file, 'r')
    X = [ map(int,line.split(',')) for line in f ]
    alpha = 0.1
    beta = 0.1

    dmm = DMM(X, int(nTopics), alpha, beta)
    dmm.dmm_learning(dmm, iteration)

if __name__ == "__main__":
    main()
```

APPENDIX C

C1. The Python code used to fit the Dirichlet Multinomial Mixture model: (Note, λ_α and λ_β are simply *alpha* and *beta* in the code.)

```
# Dirichlet Multinomial Mixture + collapsed Gibbs sampling

import numpy
from collections import Counter

class DMM:
    def __init__(self, K, alpha, beta, docs, V, smartinit=True):
        self.K = K          # number of topics
        self.alpha = alpha # parameter of topics prior
        self.beta = beta    # parameter of words prior
        self.docs = docs
        self.V = V
        self.D = len(self.docs) # number of docs in corpus
        self.theta = numpy.zeros((self.D, K)) #create array for posterior

        self.m_z = numpy.zeros(K) # document count of each topic
        self.z = numpy.zeros((self.D, K)) #topic assigment to document
        self.n_z = numpy.zeros(K)    # word count of each topic
        self.n_z_t = numpy.zeros((K, V)) # word count of each topic and vocabulary



        self.N = 0
        for m, doc in enumerate(docs):
            #sample a topic for m:
            all_z = numpy.random.multinomial(1, (1/float(self.K))*numpy.ones(K))
            z = all_z.argmax()
            self.z[m,:] = all_z
            self.m_z[z] += 1
            self.n_z[z] += len(doc)
            for t in doc:
                self.n_z_t[z, t] +=1
            self.theta[m,:] = (1/float(self.K))*numpy.ones(K)
```

```
def inference(self):
    """learning once iteration"""
    for m, doc in enumerate(self.docs):
        #record the current topic of m:
        z = self.z[m,:].argmax()

        #discount for current topic:
        self.m_z[z] -= 1
        self.n_z[z] -= len(doc)
        for t in doc:
            self.n_z_t[z, t] -=1
        #sample new topic:
        p = self.sampleTopicNo(m, doc)

        #new topic:
        all_z  = numpy.random.multinomial(1, p / p.sum())
        z = all_z.argmax()
        self.z[m,:] = all_z
        self.theta[m,:] =  p / p.sum()

        self.m_z[z] += 1

        self.n_z[z] += len(doc)
        for t in doc:
            self.n_z_t[z, t] +=1

def sampleTopicNo(self, m, doc):

    #this function calculates the probability
    p = numpy.zeros(self.K)
    overFlowCount = numpy.zeros(self.K)

    for k in range(self.K):
        p[k]= (self.m_z[k] + self.alpha) / (
        self.D - 1 + self.K*self.alpha) #first term in eq 4
        valueOfRule2 = 1.0
        counter = Counter(doc)
        i = 0
        for w in doc:
            wordFre = counter[w]
            for j in range(wordFre):
                tmp = (self.n_z_t[k, w] + self.beta + j) / (
                self.n_z[k] + self.V*self.beta + i) #second term in eq4
                i += 1
                valueOfRule2 *= tmp
                valueOfRule2, overFlowCount[k]  = self.isOverFlow(
                valueOfRule2, overFlowCount[k])

        p[k] *= valueOfRule2
```

```
            return self.reComputeProbs(p, overFlowCount)

        def isOverFlow(self, valueOfRule2, overFlowCount):
            if valueOfRule2 < 1.0e-150:
                overFlowCount -= 1
                return valueOfRule2 * 1.0e150, overFlowCount
            else:
                return valueOfRule2, overFlowCount

        def reComputeProbs(self,p, overFlowCount):

            max = overFlowCount[0];
            for k in range(self.K):
                if overFlowCount[k] > max and p[k] > 0:
                    max = overFlowCount[k];

            for k in range(self.K):
                if p[k] > 0:
                    p[k] = p[k] * numpy.power(1.0e150, overFlowCount[k] - max)

            return p

        def worddist(self):
            """get topic-word distribution"""
            return (self.n_z_t + self.beta) / (
            self.n_z[:, numpy.newaxis] + self.V*self.beta)

        def docdist(self):
            """get topic distribution"""
            return (self.m_z + self.alpha) / (self.m_z+ self.K*self.alpha)

        def perplexity(self, docs=None):
            if docs == None: docs = self.docs
            phi = self.worddist() #topic x word distribution
            log_per = 0
            N = 0
            topics = (self.n_z + self.alpha) / (
            self.n_z.sum() + self.K*self.alpha)
            for m, doc in enumerate(docs):
                log_multinomial = numpy.zeros(self.K)
                for w in doc:
                    log_multinomial += numpy.log(phi[:,w])
                log_per += (self.theta[m,:] * (
                numpy.log(topics) + log_multinomial)).sum()
                N += len(doc)
            return numpy.exp(-log_per / N)

    def dmm_learning(dmm, iteration, voca):
        pre_perp = dmm.perplexity()
```

```
    print "initial perplexity=%f" % pre_perp
    for i in range(iteration):
        dmm.inference()
        perp = dmm.perplexity()
        print "-%d p=%f" % (i + 1, perp)

    output_word_topic_dist(dmm, voca)

def output_word_topic_dist(dmm, voca):

    phi = dmm.worddist()
    #theta = dmm.theta()
    #print theta
    for k in xrange(dmm.K):
        print "\n-- topic: %d" % k
        for w in numpy.argsort(-phi[k])[:10]:
            print "%s: %f " % (voca[w], phi[k,w])



def main():

    import vocabulary
    filename = 'Data/lchf.txt'
    alpha = 0.1          #alpha hyperparameter
    beta = 0.1           #beta hyperparameter
    K = 50               #number of topics
    iteration = 15     #number of iterations
    smartinit = 'False' #smart initialize of parameters"
    stopwords = 'True'  #exclude stop words"
    df = 2               #threshold of document freaquency to cut words

    corpus = vocabulary.load_file(filename)

    voca = vocabulary.Vocabulary(stopwords)
    docs = [voca.doc_to_ids(doc) for doc in corpus]
    if df > 0: docs = voca.cut_low_freq(docs, df)

    dmm = DMM(K, alpha, beta, docs, voca.size(), smartinit)
    print "corpus=%d, words=%d, K=%d, a=%f, b=%f" % (
    len(corpus), len(voca.vocas), K, alpha,beta)

    dmm_learning(dmm, iteration, voca)

if __name__ == "__main__":
    main()
```