CHALMERS

GÖTEBORGS UNIVERSITET

MASTER'S THESIS

Multilingual Gaussian Latent Dirichlet Allocation

Elias Kamyab

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
CHALMERS UNIVERSITY OF TECHNOLOGY AND UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Thesis for the Degree of Master of Science

Multilingual Gaussian Latent Dirichlet Allocation

Elias Kamyab

CHALMERS

GÖTEBORGS UNIVERSITET

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg, Sweden
Gothenburg, September 2019

Multilingual Gaussian Latent Dirichlet Allocation
Elias Kamyab

Supervisor: Johan Jonasson, Department of Mathematical Sciences
Examiner: Richard Johansson, Department of Computer Science and Engineering

Master's Thesis 2019
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg

Multilingual Gaussian Latent Dirichlet Allocation
Elias Kamyab
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg

Abstract

Topic modeling is a type of statistical modeling for discovering the abstract "topics" that occur in corpus data. Depending on basic assumptions, there are different probabilistic algorithms for this purpose. One way to achieve this goal is to use the semantic relations between the words in the documents.

Since continuous space word embeddings or word vectors, pre-trained on a large and unstructured collection of documents, have been shown to be effective in capturing these relations, the Gaussian version of Latent Dirichlet Allocation will be the main focus in this master thesis. By this model we will be able to group words, which are a priori known to be semantically related, into topics.
To perform the inference, an introduction of collapsed Gibbs sampling, based on Cholesky decomposition of the covariance matrix of the posterior predictive, will be given.

Multilingual models are interesting in general for different reasons. In order to make the perception of the multilingual models simpler in this thesis, we call a language, where there exist possibly more data, as source language and others as targets.

We investigate how to make the Gaussian Latent Dirichlet Allocation poly- or multilingual by finding some cross-lingual bridges between languages. What is meant by multilingual (in this thesis) is that the model performs a better topic prediction on the corpus from target languages and shorten the computing time.
This will be thanks to the cross-lingual techniques that provide semantic regularities from the corpus of the source language to the target ones.
These techniques will be presented in detail in this paper in form of algorithms and the outcome and the running time from monolingual and bilingual cases will be compared to evaluate the models.

Experiments will be done on two document collections, one that contains 11,914 product reviews from the Amazon website and the other a collection of documents supplied by the Swedish audio book company Storytel. To furnish a better view of model performance, the result from Amazon data will be partly demonstrated locally in a corresponding section and the ones from Storytel's data set will be provided in the *Results and Experiments* section.
Moreover, the models will be run on both *pre-trained* and *trained* word vectors or word embeddings. In the *trained* case, the word embedding is created from the same corpus that the model later will be trained on for topic prediction.

Acknowledgements

First, I would like to take the opportunity to thank Storytel for providing the access to their large database of books and thank Arian Barakat, who patiently taught me how to use the Google Cloud Platform for this work and for other technical supervising. I surely hope that my work will be interesting to you and Storytel and will be useful in the future in some manner. Also, thanks a lot to my supervisor Johan Jonasson, who suggested the topic of this wonderful thesis and helped me with connecting to Storytel and his guidance and advice.

Elias Kamyab, Gothenburg, September 2019

List of Figures

List of Tables

Contents

1 Introduction

Nowadays, thanks to the internet, digitization and translation, the transfer of science and information worldwide is much easier and more efficient than before. As a benefit, the ability to access the same book, document, news, etc. in more than solely the original language increases. As access to the multilingual corpora grows, demand for the systems that can handle multilingual texts, in the sense that topic classification for one language should be essentially the same as for another language, increases.

Multilingual Gaussian LDA (MGLDA) is a new idea that will be investigated in this thesis. It is considered to be capable of dealing with topic modeling in a multilingual setting. In this paper, we will present and test some new ideas that make topic modeling of a corpus of a target language easier by using some information from a corpus of a source language, from which we assume that we have access to a much larger database of text. We will use some natural assumptions to make cross-lingual connection more simple and practically realistic.

Latent Dirichlet Allocation (LDA) and Gaussian Latent Dirichlet Allocation (GLDA) are Bayesian techniques that are used for topical modeling and text clustering. These models represent two different techniques that are able to capture the topic structure in a collection of documents. In this paper, we will explain these models in detail, in particular GLDA.

There are similarities between the models and there are contrasting differences as well, which partly will be mentioned in this section.
Unlike the LDA, there are some clear benefits with Gaussian LDA. In traditional LDA, a fixed vocabulary of words is assumed and the size of the vocabulary is limited to the number of unique words in the training data. Therefore it can not handle unseen words (out of vocabulary) in the *"held out documents"* (test data), but this problem is solved in Gaussian LDA. The model can assign high topic probability to an unseen word by taking advantage of the contiguity of semantically similar words in the embedding space. This makes Gaussian LDA an important and useful model in the field of Natural language processing.

There are some basic assumptions in common for both models such as *"Bag of words" (BoW)* and *"exchangeability of words and documents"* which means that the order of words in a text respectively the order of documents in a corpus can be neglected. These assumptions should not be interpreted as identically and independently distributed property. Exchangeability is rather *"conditionally* independent and identically distributed" where the conditioning is with the respect to an underlying latent parameter of a probability distribution [2]. In other words, each document is a mixture of a number of topics, and each topic has a separate *word distribution*. Depending on what these topics are, the words will be drawn conditionally from them.

The results of the topic classifications are presented in the form of top representing words, i.e. most probable words for the corresponding topic.

Both models consider a document as a mixture of a small number of topics with the difference that LDA regards the topics as discrete distributions over words whereas GLDA consider them as multivariate Gaussian distributions over the embedding space. In other words, each topic k is a multivariate normal distribution with mean μ_k and covariance Σ_k.

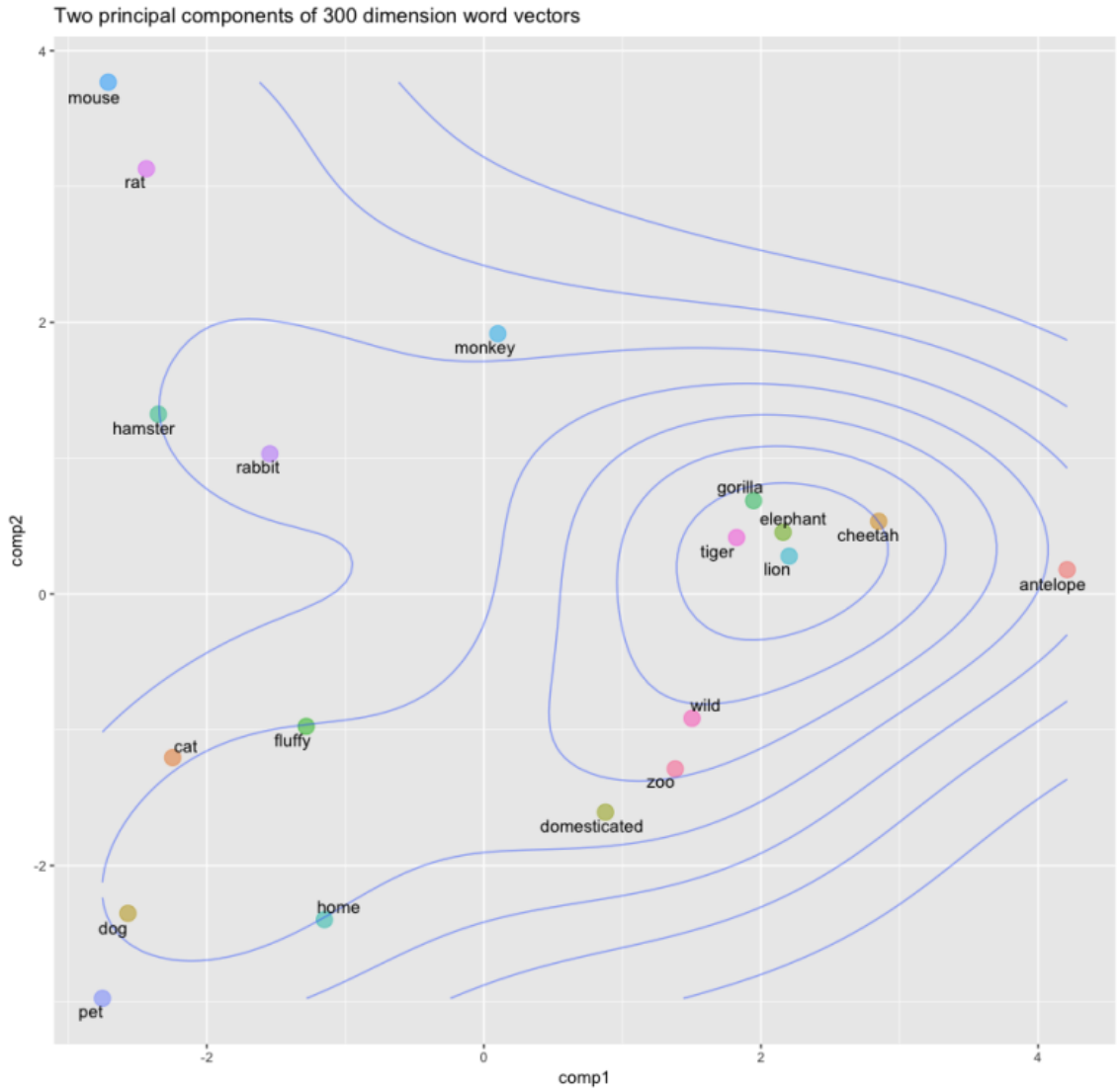
An embedding views each word as a vector of continuous numbers in a multidimensional space, named *word vector*. In this paper, the concept *word embedding* is referred to the embedding for the entire vocabulary and the *word vector* to the one for a single word.
The word embeddings are trained on very large, unstructured corpora with millions and thousands of words and documents respectively and learn syntactic and semantic coherence of the words in the sentences. They have been shown to capture lexico-semantic regularities in a language in a way that words with the similar syntactic and semantic properties are found to be close to each other in the embedding space [1]. This means, for instance, that words such as "lion" and "tiger" will have the word vectors that are very close to the word "animal", whereas the word "Mars" will be quite distant. In Figure 1.1 the proximity of the word "elephant" is shown in two dimensional space.

A beauty of representing the words as numbers in vectors is that they lend themselves to mathematical operators. For example, we can add and subtract vectors by using word vectors: king - man + woman = queen (Mikolov et al., 2013).

For simplicity in this paper, we use some abbreviations such as LDA for Latent Dirichlet Allocation, GLDA for Gaussian Latent Dirichlet Allocation and MGLDA for multilingual Gaussian Latent Dirichlet Allocation.

This thesis is organized as follows. In section 2, the LDA, Gibbs sampling, word embedding and data pre-processing will be presented. In section 3, the GLDA and all related fields such as a speeding-up technique will be discussed. In section 4, two MGLDA algorithms will be introduced and then discussed in detail. In section 5, results of experiments on both the Storytel's and Amazon data sets will be provided and the last section contains summary, calculations and openings for future work.

Figure 1.1: Two dimensional location (captured by principal component analysis) of some related words, mostly animals, illustrated in this picture. The Large animals such as "elephant", "tiger", "lion" and "gorilla" are located much closer to each other than the small ones such as "mouse" and "rat".



2 Background

In this section we will describe the LDA model, Gibbs sampling, word embedding and data pre-processing as a stepping stone towards Gaussian LDA. But first, we start with a short description of different distributions that will be used in this paper.
The Dirichlet distribution or $Dir(\alpha)$ is a multivariate generalization of the Beta distribution and is a continuous probability distribution with a positive real valued vector α as input parameter.
This distribution (like the Beta distribution) generates probability vectors. More precisely, the output $\theta \sim Dir(\alpha)$ will be an element of the simplex $\{x \in [0,1]^K; \sum_{k=1}^{K} x_k = 1\}$, where K is the dimension of α. The Dirichlet distribution is used in Bayesian statistics as a conjugate prior of the Categorical and Multinomial distributions. For a K-dimensional probability vector θ:

- A random variable X is said to be categorically distributed, $X \sim Cat(\theta)$ if $P(X = k|\theta) = \theta_k$ for each $k = 1, ..., K$.
- A random vector $Y = (Y_1, ..., Y_k)$ is multinomial distributed, $Y \sim Mult(m, \theta)$ if $X_1, X_2, ..., X_m$ are iid $Cat(\theta)$ and Y_k is the number of X_i:s such that $X_i = k$.

As mentioned before, the Dirichlet distribution is a conjugate prior of the Categorical distribution. This means that both prior and posterior have the same algebraic form. More precisely we claim that if $\theta \sim Dir(\alpha)$ and the data $X = x_1, x_2, ..., x_n$ is a vector of iid $Cat(\theta)$ random variables, the $\theta|X \sim Dir(\alpha + c)$, where c_k is the number of i such that $x_i = k$. This can be proved as follows:

$$P(X = x|\theta) = \prod_{i=1}^{n} p(x = x_i|\theta) = \prod_{i=1}^{n} \theta_{x_i}$$

Which can be rewritten as:

$$P(X = x|\theta) = \prod_{i=1}^{n}\prod_{k=1}^{K} \theta_k^{I(x_i=k)} = \prod_{k=1}^{K} \theta_k^{\sum_{i=1}^{n} I(x_i=k)} = \prod_{k=1}^{K} \theta_k^{c_k} \tag{2.1}$$

By definition:

$$p(\theta) = p(\theta|\alpha) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{K} \alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \tag{2.2}$$

By deleting the terms that do not depend on θ, we get:

$$p(\theta|\alpha) \ \propto \ \prod_{k=1}^{K} \theta_i^{\alpha_i - 1} \tag{2.3}$$

Once we have the *prior* and *likelihood*, we can find the *posterior* $p(\theta|X)$, by using *Bayes formula* as follows:

$$p(\theta|X = x) = \frac{p(X = x|\theta) \ \cdot \ p(\theta)}{p(X = x)} \propto p(X = x|\theta) \cdot p(\theta) \implies$$

$$p(\theta_k|X=x) \propto \prod_{k=1}^{K} \theta_k^{c_k} \cdot \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} = \prod_{k=1}^{K} \theta^{c_k + \alpha_k - 1} \tag{2.4}$$

This can be recognized as the $Dir(c+\alpha)$ distribution.

2.1 LDA

The LDA model regards documents as a mixture of a number of topics and topics as distributions over the words which can be interpreted that one single document can contain more than one topic and at the same time, one specific word or word type can be present in different *topics* but with different probabilities.
Once the training is done, we have estimates of the *topic distribution* of documents as well as the *word distribution* for each topic. These estimates give us a way of understanding the contents of the corpus at a high level [3].

For simplicity, some extra assumptions are made. First, the number of topics K is assumed to be known a priori. Second, the size of the *vocabulary* V (or number of unique words in the corpus) is fixed. We define the word probability matrix as $\Pi = [\pi_{kn}]$, where π_{kn} is the probability of word n for topic k. It follows from the assumptions that the dimension of Π is fixed before, namely $K \times V$.

LDA is a multi-level hierarchical Bayesian model in which the parameters α and β are corpus-level hyper-parameters. For simplicity, they are set as symmetric and uniform, i.e. $\alpha_1 = \alpha_2 = ... = \alpha_k$ and $\beta_1 = \beta_2 = ... = \beta_v$. A corpus of D documents and N_d words (in the corresponding document) is then assumed to be generated as follows:

1. Independent K-dimensional $Dir(\alpha)$ distributed topic distributions: $\theta_1, ...\theta_D$ are chosen once per document.
2. Independent V-dimensional $Dir(\beta)$ distributed word distributions: $\pi_1, ..., \pi_k$, are chosen once per topic.
3. Given θ and π, for each position $(d, n),\ d = 1, ..., D$ and $n = 1, ..., N_d$, a topic z_{dn} is chosen according to θ_d and then a word $w_{d,n}$ according to π_k and z_{dn}.

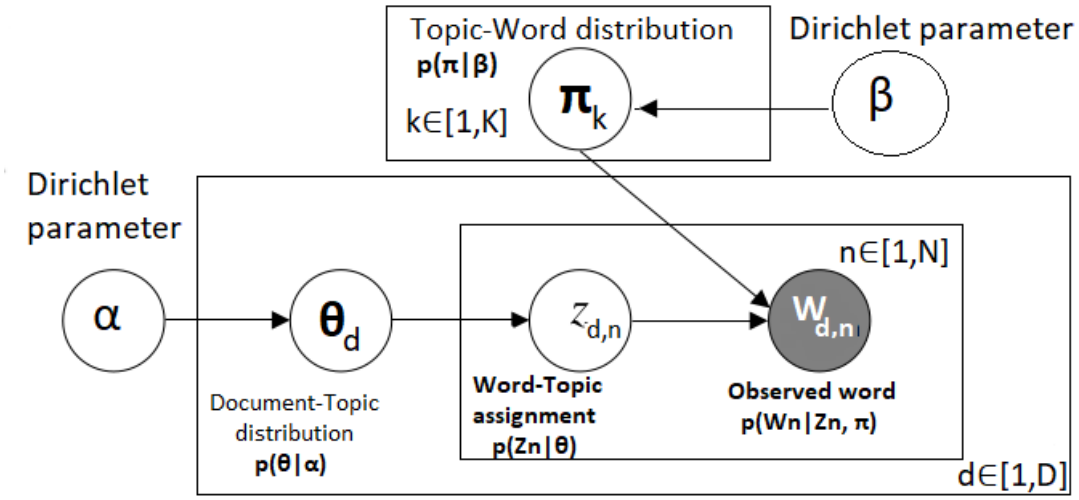


Figure 2.1: An illustration of the multi-level hierarchical Bayesian model of the LDA and how they are connected to each other.

In pseudo-code, the generative algorithm is as follow:

for *each topic* $k \in [1, K]$ **do**
  -Draw $\pi_{\mathbf{k}} \sim Dir_v(\beta)$
end
for *each document* $d \in [1, D]$ **do**
  -Draw $\theta_d \sim Dir_k(\alpha)$
  for *each word* $n \in [1, N_d]$ **do**
    -Draw a topic $z_{d,n} \sim Cat(\theta_d)$
    -Draw a word (word index) $w_{d,n} \sim Cat(\pi_{z_{d,n}})$
  end
end

Algorithm 2.1: Generative process of the LDA

LDA is a Bayesian model and we want to find the posterior distribution over the topic parameters and latent topics. At the same time we realize that the calculation of this posterior (the joint distribution below)

$$p(\theta, \pi, \mathbf{Z}|\mathbf{W}) = \frac{p(\theta, \pi, \mathbf{Z}, \mathbf{W})}{p(\mathbf{W})} \tag{2.5}$$

is intractable due to the denominator in the right hand side. The equation for the denominator is as follow:

$$p(\mathbf{W}) = \left(\prod_{k=1}^{K} \int p(\theta_k|\alpha) p(\pi_k|\beta) \right) \left(\prod_{d_n=1}^{N_d} \sum_{z_{d_n}} p(z_{d_n}|\theta) p(W_{d_n}|Z_{d_n}, \pi) \right) d\theta_1 ... d\theta_k \, d\pi_1 ... d\pi_k \tag{2.6}$$

The function above can be rewritten as:

$$p(\mathbf{W}) = \left(\frac{\Gamma(\sum \alpha_k)}{\prod \Gamma(\alpha_k)} \frac{\Gamma(\sum \beta_k)}{\prod \Gamma(\beta_k)} \int \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \pi_k^{\beta_k - 1} \right) \left(\prod_{n=1}^{N_d} \sum_{k=1}^{K} \prod_{v=1}^{V} (\theta_k \pi_{k,v})^{w_n^v} \right) d\theta_1 ... d\theta_k \, d\pi_1 ... d\pi_k \tag{2.7}$$

Obviously it is intractable to compute this.

As an approximate computation of the posterior, there are some inference algorithms such as Variational inference and Markov Chain Monte Carlo, the latter one usually in the form of Gibbs sampling. In this thesis we will focus on Gibbs sampling.
In the next subsection we will briefly introduce Gibbs sampling and how this technique can be used for the LDA model.

2.2 Collapsed Gibbs sampling for LDA

Gibbs sampling is a special case of Markov Chain Monte Carlo (MCMC). In MCMC as for any Markov chain, the state at each time step, conditionally given the previous step, is independent of the states at times before that.
Gibbs sampling is a popular technique for generating random samples of multivariate data from complex distributions. The sampling technique works by at each time step picking a coordinate at random and then updating the value there according to the conditional distribution given the values at all other coordinates.

In the case of the LDA model, we sample from the posterior distribution of the latent topics given the corpus. This is done by picking a document d and word n at random from the corpus and update $z_{d,n}$ according to:
$p(z_{d,n} = k|z_{\neg(d,n)}, V, \alpha, \beta),\ k = 1, ..., K.$

There are two kind of counts that have been used in this sampling, $n_{d,k}$ and $n_{k,v}$. The first one represents the number of words assigned to topic k across a document and the latter one the number of positions a topic k assigned to word v across a vocabulary.
Once a topic for the current word is sampled, the $n_{d,k}$ and $n_{k,v}$ will be updated, which in turn will be used for the next step of the MCMC. This sampling distribution will converge to the true distribution in the long enough Gibbs-iterations. The Gibbs algorithm in this form is called *collapsed* Gibbs due to the fact that we integrate out or collapse some parameters of the true posterior, in this case θ and π.

By the above, we sample from the posterior probability distribution of topic assignment:

$$p(\mathbf{Z}|\mathbf{W}, \theta, \pi, \alpha, \beta) = \frac{p(\mathbf{Z}, \mathbf{W}|\theta, \pi, \alpha, \beta)}{p(\mathbf{W}|\alpha, \beta)} \tag{2.8}$$

The Equation 2.8 can be rewritten (after collapsing) as below:

$$p(z_{d,n} = k|z_{\neg(d,n)}, V, \alpha, \beta) \ \propto \ \left(\frac{\alpha \ + \ n_{d,k}}{K\alpha \ + \ \sum_{k=1}^{K} n_{d,k}}\right) \cdot \left(\frac{\beta \ + \ n_{k,v}}{V\beta \ + \sum_{v=1}^{V} n_{k,v}}\right) \tag{2.9}$$

By formula (2.9), the training algorithm for the LDA with collapsed Gibbs sampling can be defined as follows:

-Initialize the topic of each word randomly from $\sim U(1, K)$
for *each iteration* $i \in [1, I]$ **do**
  for *each document* $d \in [1, D]$ **do**
    for *each word* $n \in [1, N_d]$ **do**
      for *each topic* $k \in [1, K]$ **do**
        -Calculate probability of topic k by formula:
        $p(z_{d,n} = k | z_{\neg(d,n)},\ V,\ \alpha,\ \beta) = \left(\frac{\alpha + n_{d,k}}{K\alpha + \sum_{k=1}^{K} n_{d,k}}\right) \cdot \left(\frac{\beta + n_{k,v}}{V\beta + \sum_{v=1}^{V} n_{k,v}}\right)$
      end
      - Normalize the probability vector $\mathbf{p}$=$[p_1, p_2, ..., p_k]$ to sum of one.
      - Sample a new topic k from $\mathbf{p}$ and assign it to the current word $z_{d,n}$
    end
  end
  - (Optional) Calculate perplexity after each iteration
end

Algorithm 2.2: LDA with collapsed Gibbs sampling

Perplexity is a measure of how well a discrete probability distribution predicts samples and is algebraically equivalent to the inverse of the geometric mean per-word likelihood or average of *negative log likelihood*.
This measurement applies on test data or data that the model has not been trained on and is supposed to have a monotonically decreasing trend [2]. A low perplexity indicates that the model is good at predicting and has a good generalization performance.

The formula for the perplexity in the LDA is as follows:

$$\mathbf{perplexity}(Data_{test}) = exp\left(-\sum_{d=1}^{D} \frac{\sum_{n=1}^{N_d} \log p(W_n)}{N_d}\right) \tag{2.10}$$

In Figure 2.2, the perplexity is shown as a function of the number of Gibbs iterations. The LDA model is run on 10% of Amazon reviews, of which 90% for training the model and 10% held out for the testing purpose. The corresponding results in terms of top words for topics are presented in the Table 5.1 in *Results and Experiments* section.

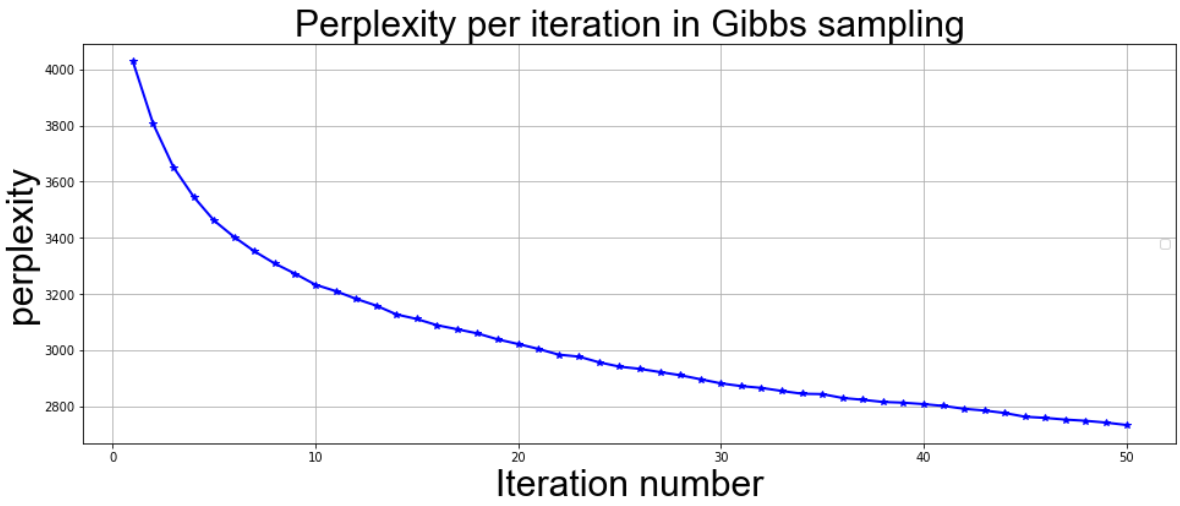


Figure 2.2: An illustration of the perplexity values over 50 iterations on the test data set. The decreasing trend of the curve means that the results of the LDA improve for each Gibbs iteration.

2.3 Word embedding

In this part, the process of representing words as real vectors will be outlined. The word embedding represent each word w as a vector $\mathbf{v}(w) \in \mathbb{R}^M$ for some given dimension M. The idea behind word embedding is that each unique word in the vocabulary will have its own unique vector, but words with a close semantic and syntactic relation will be located close to each other in that multidimensional space.
As a side remark, we note that it has been found, sometimes, that similarity of word representations goes beyond syntactic regularities and even some mathematical operations can be applied to the embedding. It was shown, for example, that vector("King") - vector("Man") + vector("Woman") results in a vector that is closest to the vector representation of the word "Queen" [1].

How are these word embeddings generated? There are two traditional algorithms for this purpose, *continuous bag of words (CBOW)* and *continuous Skip-gram*. CBOW predicts the current word based on the context, whereas the skip-gram predicts the context given the word. Hence CBOW and skip-gram can be considered as inverted methods of each other. The idea is to use the fact that the words that share similar contexts tend to have similar meanings.

How do these work? Assume each word in the vocabulary is indexed by $i = \{1, 2, ..., V\}$, where V is the vocabulary size. Take a word as central word w_c and the surrounding words in a size of m (max window size) as context words: $w_{s1}, ..., w_{s2m}$.
In order to compute the above-mentioned probabilities, each word in CBOW and Skip-gram is represented as two vectors:

- $\mathbf{v}_i \in \mathbb{R}^M$ when the word at index i is a central word.
- $\mathbf{u}_i \in \mathbb{R}^M$ when the same word is a context (surrounding) word of other words that now are central words.

The conditional probability of generating a central word given the context can be defined as follow:

$$P(w_c|w_{s1}, ..., w_{s2m}) = \frac{e^{\mathbf{u}_c^T \frac{(\mathbf{v}_{s1}+...+\mathbf{v}_{s2m})}{2m}}}{\sum_i e^{\mathbf{u}_i^T \frac{(\mathbf{v}_{s1}+...+\mathbf{v}_{s2m})}{2m}}}$$

In skip-gram case, under the assumption that context words given a central are independently generated, the conditional probability (in basic formulation by Softmax) is obtained as follows [5]:

$$P(w_s|w_c) = \frac{e^{\mathbf{u}_s^T \mathbf{v}_c}}{\sum_i e^{\mathbf{u}_i^T \mathbf{v}_c}}$$

where $\mathbf{u}_s^T \mathbf{v}_c$ is the inner product of the vectors.

Something important to pay attention to is that the word embedding is only a *byproduct* of a predictive task and not its output. The output is, as we just stated,

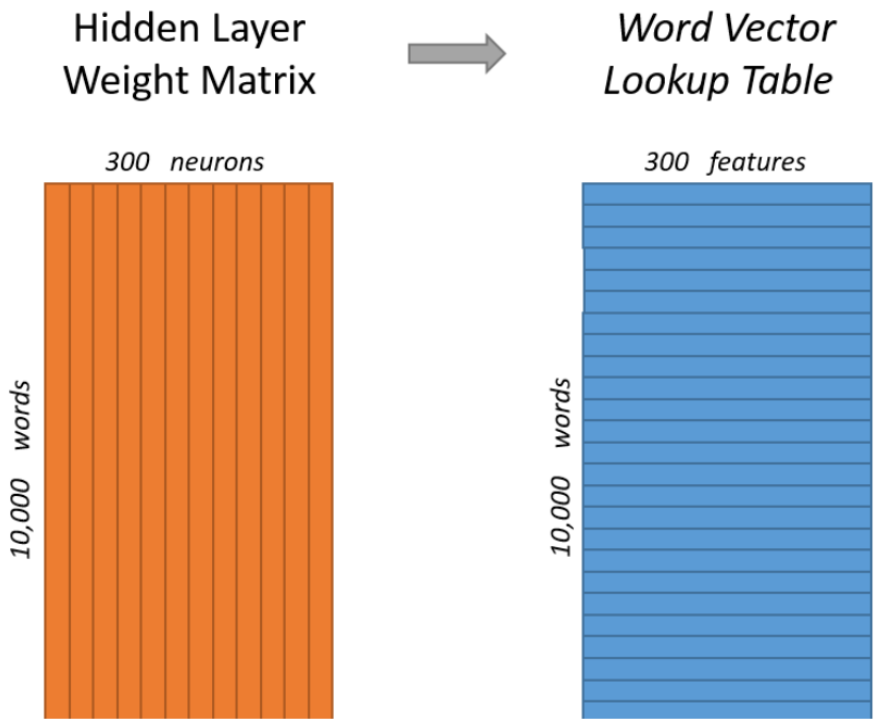
word- or context-probability given the context or the word respectively.

The embedding of a given word is then given as a function of the weights of the hidden layer, where these have been chosen to minimize the loss function of the output predictions. The loss (negative log likelihood) for skip-gram is given as follows:

$$-\log P(w_s|w_c) = -\mathbf{u}_s^T \mathbf{v}_c + \log\left(\sum_i e^{(\mathbf{u}_i^T \mathbf{v}_c)}\right) \tag{2.11}$$

By representing words as one-hot vectors, the weights of the hidden layer can be seen as the dimensions of the word vector where each row represents a word and each column (or neuron in the hidden layer) represents a dimension.

Figure 2.3: There are 10000 words in a toy corpora and we want to build an 300 dimensional word embedding. Each neuron in the hidden layer represents a dimension in the weight vector (word vector)



In general, the Softmax formulation is impractical since the computational complexity of gradient of the loss is proportional to the vocabulary size.

In order to reduce this complexity, an approximate training method called negative sampling is introduced [6]. In this case, all context words are considered as positive examples and negatives or noise words are sampled at random from the dictionary (vocabulary).
Now the problem is considered as a binary classification task since the goal is to independently predict the presence or absence of the context word. In other

words, to distinguish a context word w_s from draws from the *noise distribution*. The Softmax loss in Equation 2.11 replaces by binary logistic loss function in the Equation below:

$$-\log P(w_s|w_c) = -\log\left(\sigma(\mathbf{u}_s^T\mathbf{v}_c)\right) - \sum_{n}^{\mathbb{N}} \log\left(\sigma(-\mathbf{u}_n^T\mathbf{v}_c)\right) \qquad (2.12)$$

where $\sigma(\mathbf{u^T}\ \mathbf{v}) = (1 + e^{-u^T\ v})^{-1}$ and $\mathbb{N}$ is number of negative words.
In negative sampling, the gradient computation in each step of the training is no longer related to the vocabulary size, but linearly related to $\mathbb{N}$.

Let's explain the skip-gram process by a simple example. Suppose we have this text as a toy text: "my cat chases mice in the garden at nights". The n-gram training sample for n = 5, window = 2 and "chases" as central word will be as follows:

(*my, cat, chases*), (*my, cat, chases, mice*), (*my, cat, chases, mice, in*)

(*cat, chases, mice, in*), (*chases, mice, in*)

The training samples (the surrounding words of the central word) include only the words in max range of 2 $\times$ window, not all the words in the text. This means (in the simple example above) given the central word "chases", the words "cat" and "mice" will have a higher probability estimate than the words "my" and "in" and the remaining words will have zero probability.

CBOW and Skip-gram have some advantages and disadvantages compared to each other. Since CBOW can use many context words to predict a single target word, it offers very good performance even when our input data is not so large. It trains several times faster than Skip-gram, has a better accuracy for frequent words and does not take so much memory.
On the other hand, Skip-gram works well with a larger amounts of data and has better accuracy for rare words. The skip-gram model is able to select more information from training data and consequently offer more accurate embeddings when the training data is large, but training takes longer time and needs more memory than CBOW.

Both algorithms use local context with a defined number of neighboring words or window. These methods are used by *Google's Word2Vec* and are expended as basic methods for other embedding algorithms. One can say that, the word embedding of type *Word2vec* is a combination of CBOW and Skip-gram methods.
There are other and newer embedding algorithms as well, such as *Global Vectors for Word Representation* or GloVe and *Facebook's Fasttext*.

The GloVe algorithm is introduced by *Pennington, Socher and Manning (2014)* [7] and can be seen as an extension of *word2vec*. It seems to have a better performance by capturing some semantic relations such as analogous properties

between the words. For example, the analogy "king to queen as man to woman" should be encoded in the vector space by the vector equation: king - queen = man - woman [8]. This semantic improvement of the algorithm is due to the fact that the method takes advantage of global co-occurrence counts, and statistics of the whole corpus rather than the local context or local co-occurrence.

None of the mentioned methods above can handle *out of vocabulary words* in the sense that an unseen word in the training corpus will get a word vector. But, by incorporating sub-word information, the *Fasttext* algorithm has the ability to support unseen words. The algorithm splits all words into a bag of n-gram characters usually of size 3-6. This sub-word information is then added up to create a new word as a final feature. This makes the *Fasttext* a powerful embedding method. Due to this fact, we will mainly focus on this method for the rest of this thesis.

When doing topic modeling, we can either choose to use a *pre-trained* word embedding or train one ourselves from the text data relevant to our application.

2.3.1 Pre-trained word embedding

There are some *pre-trained* word vectors by Facebook[1], Stanford [2] and Google. Some exist in different languages and some mainly in English with many borrowed non-English words from other languages. They have been trained on very large and unstructured corpora of billions of words. The larger corpus the word embedding has been trained on, the higher the semantic accuracy and the more comprehensive the embedding will be.

Pre-trained word embeddings are of different dimensions. The higher the dimension the more information can be captured from the corpus and consequently the higher accuracy will be achieved when using it in the GLDA model, but of course it requires longer training time and also a larger database of texts to train on. The most common dimension of the current *pre-trained* word vector are 300.

Table 2.1: 3 types of *pre-trained* word vectors and some related informations

Company or Author:	Facebook	Stanford	Google
Embeddings name:	FastText	GloVe	Word2Vec
Algorithm behind:	bag of n-gram characters	Global statistic's and co-occurence	CBOW & skip-gram
Dimension:	300	25,50,100,200,300,1000	300, 1000
Language:	294 languages	English	English

Some of the word embeddings above are very memory consuming, up to 5-6 gigabytes and take time to download. In Table 5.2 in the *Results and Experiments* section, a *pre-trained* word embedding from Fasttext with dimension of 300 and

[1] `https://fasttext.cc/docs/en/crawl-vectors.html`
[2] `https://developer.syn.co.in/tutorial/bot/oscova/pretrained-vectors.html`

vocabulary size of 1 million words (wiki.news.300d.1M.vec) has been used for training the GLDA.

2.3.2 Training your own word embedding

It is possible to train your own word embedding on your own database. Most of the *pre-trained* word embeddings are trained on Wikipedia, Common Crawl, Twitter or text from social media, which might be considered as an advantage since we can use them generally for any kind of text and in any field.
However this can be seen as a *drawback* too partly because most of this heavy volume of vectors will not be used on the data you want to classify. As a consequence, this will slow down the training since the dictionary of embedded words is much larger than necessary. Therefore, it can be a good idea to train your own embedding.

Facebook has created a library that called *Fasttext* that can be imported from the Gensim package in Python. Using this, one can train one's own word embedding on text that better represents to context of the corpus that one wants to train a model on. There are some hyper-parameters in the library that need be adjusted before training while others can be left as default or optional. The choice of these parameters has a large effect on the embeddings and consequently on the results of topic classification. Some of the important parameters will be introduced here[3].

- **size** or dimensionality of word vector (M): For larger training data with a huge vocabulary size, we need higher dimensions. High dimensionality requires longer time and harder training. The range between 100 and 300 is popular. The default dimension is 100.

- **learning rate** or α: The higher the α, the faster the model converges to a solution, but also higher risk of overfitting. Another risk is that the model might miss the global minimum of the loss. On the other hand, having too low α can lead to getting stuck in local minima. The default rate is 0.05.

- **sg = {1,0}**: zero for choosing *CBOW* algorithm as underlying algorithm and one for *continuous skip-gram*.

- **word-ngram = {1,0}** zero for *Word2Vec* algorithm and one for *Fasttext* and enriching the word embedding with sub-word or char (n-grams) information.

- **min-n**: Minimum length of char n-grams to be used for training word representations.

- **max-n**: Max length of char n-grams to be used for training word representations. The higher the amount of n-grams, the more information about the local word order will be included. The values 3 and 6 are popular as min-

[3] `https://radimrehurek.com/gensim/models/fasttext.html`

and max-n.

- **window**: The maximum distance between the input and predicted word within a sentence. 2 $\times$ window defines the maximum number of surrounding words of the current word in a training sample. Larger window size enriches the embeddings with more topical information and lower with more semantical.

Once you get familiar with these hyper-parameters and have a sufficiently large training corpus, you can train your own word embedding. Training your own word embedding on a smaller corpus spares you time and memory, but such embedding are not useful for other kinds of text data.

In Table 5.3 in *Results and Experiments* section, a word embedding is first trained on the entire Amazon data set by hyper-parameters: $M = 150$, word-ngram = 1, $\alpha = 0.05$, 3 as min-n and 5 as max-n and then the embedding is used as word vectors in the GLDA.

2.4 Data Pre-processing

Before starting with the training, we need to prepare the data. Texts and corpus are written in natural language for human perception, But in text mining, those data are not always easy for computers to process. [12]

Tokenization and filtering are among the important pre-processing steps of a text. The first one is about treating a text as a string and partition it into a list of tokens and the second one about cleaning.
Cleaning or filtering a corpus (from noisy words) helps us partly to save memory and computation time and partly to give more weight to the words that are topic informative. [12] The steps in filtering are as follows:

- removing stopword or highly frequent words such as "the", "of", "in" and "on".
- removing characters such as #, ¤ and &.
- deleting numbers.
- converting capitals into lowercase.
- removing extremely short words such as "I", "u" and "r".
- getting rid of misspelled words.

An example of a text from Amazon reviews before and after filtering is present as below:

Before filtering: outstanding camera in every respect <3.. i 've owned several digital cameras in recent years (sony &nikon). the n2 's giant 3" lcd, the touch screen features,
After filtering: outstanding camera every respect owned several digital cameras recent years sony nikon giant lcd touch screen features

The tokenized text: 'outstanding', 'camera', 'every', 'respect', 'owned', 'several', 'digital', 'cameras', 'recent', 'years', 'sony', 'nikon', 'giant', 'lcd', 'touch', 'screen', 'features'

3 GLDA

Gaussian latent Dirichlet allocation or GLDA was introduced by Das, Zaheer and Dyer [3]. As mentioned earlier, the model shares some basic premises with the LDA such as the assumption of *exchangeability* and considering a document as a mixture of topics and topics as distributions over the words. But unlike LDA, the GLDA considers topics as multivariate Gaussian distributions on the continuous embedding space rather than discrete categorical distributions over words.

In other words, each topic k is a Gaussian distribution with expectation μ_k and covariance matrix Σ_k on the embedding space $\mathbb{R}^M$. This is to say that a word vector chosen from topic k is drawn from $N(\mu_k,\ \Sigma_k)$.
The conjugate priors of μ_k and Σ_k are as below:

- an *Inverse Wishart distribution* for the covariance $\Sigma_k \sim W^{-1}(\Psi, \nu)$, where $\nu \geq M$ is the prior of degrees of freedom and $\Psi \geq 0$ is a scale positive definite matrix of size $M \times M$.
- Given Σ_k, a zero centered Gaussian distribution for the mean $\mu_k \sim (\mathbf{0}, \frac{\Sigma_k}{\kappa})$ with the prior covariance (scaled by a real scalar $\kappa > 0$) as its covariance.

Since we don't have any prior knowledge of K-dimensional dirichlet parameter α, a symmetric and uniform one will be considered. Therefore, the distribution can be parametrized by a single scalar α, i.e. $\alpha_1 = ... = \alpha_k$. We set $\mathbf{\Psi} = 3 \cdot \mathbb{I}_{M \times M}$ and $\nu = M + 1$.

The generative process of the GLDA algorithm will be as follows:

for *each topic* $k \in [1, K]$ **do**
  - Draw prior topic covariance $\Sigma_k \sim W^{-1}(\Psi, \nu)$
  - Draw prior topic mean $\mu_k \sim N(\mathbf{0}, \frac{\Sigma_k}{\kappa})$
end
for *each document* $d \in [1, D]$ **do**
  -Draw topic distribution $\theta_d \sim Dir_k(\alpha)$;
  for *each word* $n \in [1, N_d]$ **do**
    -Draw a topic $z_n \sim Cat(\theta_d)$
    -Draw a word (word vector) $\mathrm{v}_{d,n} \sim N(\mu_{z_n}\ ,\ \Sigma_{z_n})$
  end
end

Algorithm 3.1: Generative process of GLDA

As for the LDA, computing the true posterior is infeasible and again we will instead use Gibbs sampling which will be discussed in the next subsection.

3.1 Collapsed Gibbs sampling for GLDA

The desire is to infer a posterior distribution for the topic parameters $\mu_{\mathbf{k}}, \mathbf{\Sigma_k}$ and the topic assignment of individual word $z_{d,n}$. Since the true posterior is analytically unsolvable and thanks to the fact that some variables can be collapsed or integrated out, a collapsed Gibbs sampler has been derived [4]. The equation below is the conditional distribution of the topic of the current word.

$$P(z_{d,n} = k|z_{\neg(d,n)}, \mathbf{V}_d, \zeta, \alpha) \ \propto \ (n_{d,k} + \alpha) \cdot t_{df_k}\left(\mathbf{v}_{d,n}|\ \mu_{\mathbf{k}}, \frac{\kappa_k + 1}{\kappa_k}\Sigma_k\right) \tag{3.1}$$

Here:

- $z_{\neg(d,n)}$ represents the topic assignments of all words in the corpus, except the one at n^{th} position of document d.
- $\mathbf{V_d}$ is the sequence of word vectors for the words in document d.
- The tuple $\zeta = (\mathbf{\Psi}, \nu, \kappa)$ represents the parameters of the prior distribution.
- α is learning rate.
- $n_{d,k}$ represents the count of words assigned to topic k in each document.
- $\kappa_k = \kappa + \sum_1^D n_{d,k}$ is scalar and a scale factor for the mean and covariance.
- $\nu_k = \nu + \sum_1^D n_{d,k}$ is scalar and a scale factor to calculate the degrees of freedom $df_k = \nu_k - M + 1$.
- $t_{df}(\mathbf{x}'|\mu', \mathbf{\Sigma}')$ is the density of multivariate t-distribution (or Student distribution) with degrees of freedom df, location $\mu^{'}$ and shape $\Sigma^{'}$.

By replacing the second term of the Equation 3.1 to PDF of the multivariate t-distribution[4], the equation can be written as follows:

$$p(z_{d,n} = k|z_{\neg(d,n)}, \mathbf{V}_d, \zeta, \alpha) \propto (n_{d,k}+\alpha)\cdot\frac{\Gamma[\frac{df_k+M}{2}]}{\Gamma(\frac{df_k}{2})df_k^{\frac{M}{2}}\pi^{\frac{M}{2}}|\Sigma_k|^{-1}}\left(1 + \frac{(\mathbf{x} - \mu_{\mathbf{k}})^T\Sigma_k^{-1}(\mathbf{x} - \mu_{\mathbf{k}})}{df_k}\right)^{-\frac{df_k+M}{2}} \tag{3.2}$$

where $\mathbf{x}$ is the word vector for the word in position d, n.
Topic parameters μ_k and Σ_k, where k is the topic of the current word x_k, updates twice for each word, once before the assignment of the new topic (by formula 3.2) and once after. The formulas for corresponding update are as follow:

$$\Sigma_k^{new} := \Sigma_k^{old} - zz^T, \ \textit{where}\ z = (x_k - \mu_k^{old})\cdot\sqrt{\frac{\kappa_k + 1}{\kappa_k}}\ \textit{and}\ \mu_k^{new} := \frac{\mu_k^{old}\cdot(\kappa_k + 1) - x_k}{\kappa_k} \tag{3.3}$$

$$\mu_k^{new} := \frac{\mu_k^{old}\cdot(\kappa_k - 1) + x_k}{\kappa_k}\ \textit{and}\ \Sigma_k^{new} := \Sigma_k^{old} + zz^T, \ \textit{where}\ z = (x_k - \mu_k^{new})\cdot\sqrt{\frac{\kappa_k}{\kappa_k - 1}} \tag{3.4}$$

The training algorithm of the Gibbs sampling can be defined as follows:

[4] https://www.researchgate.net/publication/228613894_A_short_review_of_multivariate_t-distribution

-Initialize the topic of each word randomly from $\sim U(1, K)$
for *each iteration:* $i \in [1, I]$ **do**
  for *each document in the corpus:* $d \in [1, D]$ **do**
    for *each word in the document:* $n \in [1, N_d]$ **do**
      - $n[d, k] = n[d, k] - 1$, k = topic of the current word,
      - Update topic parameters: $\mathbf{\Sigma_k}, \mu_\mathbf{k}$ by formula 3.3
      for *each topic* $k \in [1, K]$ **do**
        -Calculate probability: $p(z_{d,n} = k | z_{\neg(d,i)}, \mathbf{V}_d, \zeta, \alpha) =$ by Equation 3.2
      end
      - Normalize the probability vector $\mathbf{p}_{d,n} = [p_1, p_2, ..., p_k]$ to sum of one.
      - Sample a topic k from $\mathbf{p}$ and assign it as new topic to the current word
      - $n[d, k] = n[d, k] + 1$
      - Update topic parameters: $\mu_\mathbf{k}, \mathbf{\Sigma_k}$ by formula 3.4
    end
  end
  - (optional) Calculate the loss function after each iteration
end

Algorithm 3.2: Training algorithm for GLDA with Gibbs sampling

A low *loss function* (in our case *Negative log-likelihood*) indicates a better optimization of the parameters. The formula for the average of the negative log-likelihood is given by:

$$\text{mean}(-\log\ P(\mathbf{X}|\mu, \mathbf{\Sigma})) = \frac{\sum_d^D \frac{\sum_n^{N_d} -\log\left((2\pi)^{-\frac{M}{2}}\ Det(\Sigma_k)^{-\frac{1}{2}}\ e^{-\frac{(\mathbf{x_{d,n}}-\mu_\mathbf{k})^T \Sigma_k^{-1}(\mathbf{x_{d,n}}-\mu_\mathbf{k})}{2}}\right)}{N_d}}{D} \tag{3.5}$$

In Figure 3.1, Figure 3.2 and Table 3.1, the algorithm with Gibbs sampling of 15 iterations and 50 topics K is run on 10% of Amazon data (consisting of 1191 reviews and 75616 words) with *pre-trained* word embedding (of dimension M equal to 300) named *wiki-news-300d-1M*.
Figure 3.1 visualizes the monotonous behavior of the loss function. We can clearly see that the loss decreases as the optimization progresses. As a consequence, the updating rate in Figure 3.2, where the new topic of the current word is unequal to its old topic, decreases.

Figure 3.1: A decreasing trend of negative log-likelihood. The picture might indicate that a good number of Gibbs iterations is 8 due to the minimum point

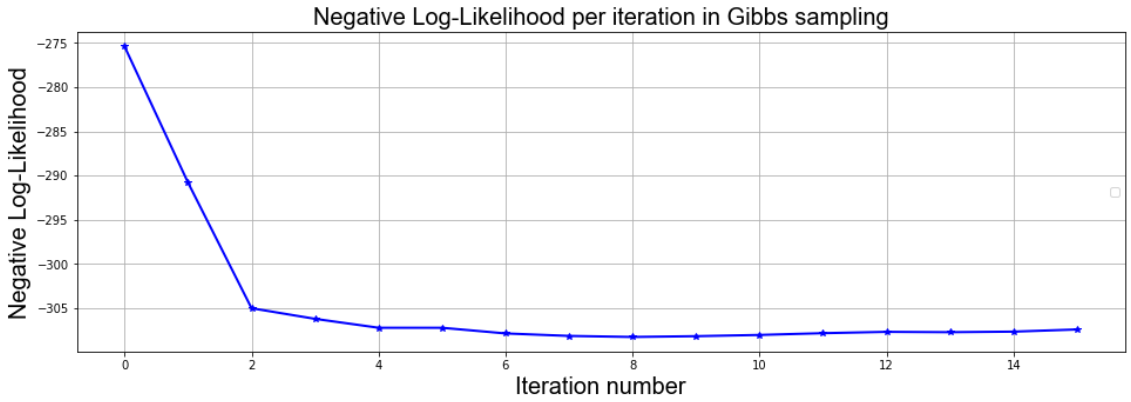
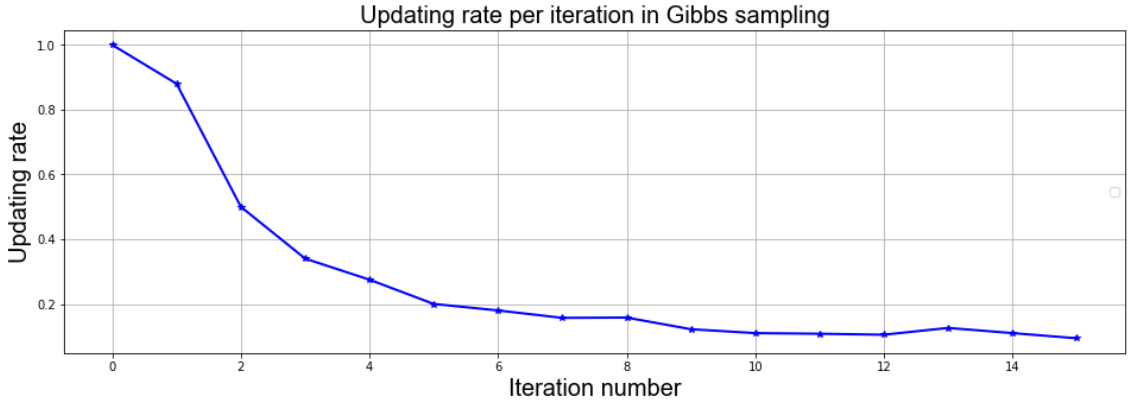


Figure 3.2: While the Gibbs algorithm learns in each iteration, the updating rate, i.e. the number of words that switch topic assignment between each iteration decreases.



In Table 3.1, total number of words assigned to each topic in iteration number = [0, 1, 5, 10, 15], are illustrated. Initially (in iteration number 0), a topic $\in [1, K]$ is assigned uniformly to each word. Then in each iteration, the sampling converges more and more to some specific topics and finally in the last iteration, to 26 topics.

Table 3.1: Topic assignment of some iteration by Gibbs sampling in GLDA. By document-wise sum of the counts $n_{d,k}$ one can notice how fast the topics converge to some specific topics

iter. No	The	number	of	words	assigned	to	each	topic		
0	1954	1632	1589	1063	1694	1303	1017	1188	1759	1570
	1784	1383	1794	1842	1833	1688	1578	1790	1585	1554
	1207	1499	1482	1370	1471	1548	1304	1195	1406	1133
	1860	1989	1231	1235	1385	1397	1181	1451	1294	1718
	1578	1234	1429	2036	1874	1755	1509	1585	1410	1250
1	786	2709	313	296	3071	172	3594	3368	12911	55
	321	708	4015	4436	795	3466	1277	1155	95	111
	222	2087	149	109	315	128	1987	248	360	728
	5232	1529	50	1164	191	65	322	1308	2201	175
	234	105	972	4028	2507	905	1886	2253	306	196
5	1443	2757	0	0	2555	0	2074	1904	3317	0
	0	4204	2434	3055	0	2822	865	3152	0	0
	0	3418	0	0	0	0	2874	0	0	10847
	2432	1276	0	7584	0	0	0	1933	2352	0
	0	0	1863	1545	1922	3352	1451	2185	0	0
10	10176	2372	0	0	2215	0	3750	1856	3334	0
	0	2090	2313	2812	0	2679	1035	2540	0	0
	0	3471	0	0	0	0	6561	0	0	3248
	2399	1352	0	2832	0	0	0	2120	2261	0
	0	0	4664	1478	1918	2712	1675	1753	0	0
15	3850	1736	0	0	2272	0	4195	1799	2631	0
	0	2389	2221	2524	0	2846	9574	2748	0	0
	0	3031	0	0	0	0	2612	0	0	3402
	1960	1376	0	2377	0	0	0	6037	3341	0
	0	0	3014	1999	1753	2558	1614	1757	0	0

By comparing the output of the GLDA to the LDA as will be seen later in *Results and Experiments* section, one can say that the GLDA model seems to produce more comprehensible topics than LDA. However, it comes at a cost; the running time for the algorithm is much higher than for LDA. This depends partly on the high dimensionality of the word embedding and partly, in particular in the pre-trained case, on the vocabulary size.

In the next subsection, a technique that will shorten the computational time and save memory will be explained.

3.2 Speed Up

As can be seen in Algorithm 3.2 and Equation 3.2, for computation of the posterior predictive we need to calculate the determinant $|\Sigma_k|$ and the inverse of the posterior covariance matrix Σ_k^{-1}. The naive computation of these terms cost $O(M^3)$ operations [3]. Since the count matrix $n_{d,k}$ and the scale parameters ν_k, κ_k change and update during the topic assignment, the determinant and the covariance need to be calculated at least as many times as the number of words $\times$ number of topics $\times$ number of Gibbs iterations. This computation can take up to several days for a medium to a large size corpus. To shorten the required running time, a technique called Cholesky factorization, will be applied on the covariance.

3.2.1 Cholesky factorization

Depending on the characteristics of a matrix, there are different variants of matrix factorization that can be employed with the purpose of saving memory and computational time. As in Das et al, we use Cholesky, or LDL^T, factorization

Assume that the matrix A is symmetric. One can write A as $A = LU$, where the upper triangular matrix U can be constructed by Gaussian elimination and the lower triangular L by row operation. Taking $D = diag(U)$, we get $U = DX$, where X is uniquely determined and has ones on its diagonal. Since A is symmetric and L^T has ones on the diagonal, $X = L^T$ and hence $A = LDL^T$. If A is positive definite $D^{1/2}$ is defined and we can write $A = CC^T$, where $C = LD^{1/2}$. This is the so called Cholesky factorization.

An example of the Cholesky factorization is as follows:

$$A = \begin{bmatrix} 4 & 8 \\ 8 & 25 \end{bmatrix}, \textit{then}:$$

$$\underbrace{\begin{bmatrix} 4 & 8 \\ 8 & 25 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} * \underbrace{\begin{bmatrix} 4 & 8 \\ 0 & 9 \end{bmatrix}}_{U} = \underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} * \underbrace{\begin{bmatrix} 4 & 0 \\ 0 & 9 \end{bmatrix}}_{D} * \underbrace{\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}}_{L^T}$$

LDL^T *can be written as:* $LD^{\frac{1}{2}}D^{\frac{1}{2}}L^T$

$$A = \underbrace{\left(\underbrace{\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}}_{D^{\frac{1}{2}}} \right)}_{C} \underbrace{\left(\underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}}_{D^{\frac{1}{2}}} \underbrace{\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}}_{L^T} \right)}_{C^T} = CC^T \tag{3.6}$$

Note that $diag(U)$ has positive elements only if A is positive definite and it is only then that one can write $C = LD^{1/2}$.

Why is Cholesky decomposition so useful? The importance appears when we estimate how much each operation might cost. As mentioned in the paper [3] if we don't use Cholesky factorization, a solution to the equation $Ax_k = b_k$ will in general cost $O(M^3)$ operations (addition and multiplication). Given a LU factorization, the cost reduces to $O(M^2)$ since $Lz = b$ and $Ux = z$ both cost $O(M^2)$.
Now the LU costs $O(M^3)$ in itself, but it can be done once and for all. Once we have Cholesky factorization $\Sigma_k = C_k C_k^T$, we can in future work solely with C_k instead of Σ_k, in the Equation 3.2 and 3.5.

The expression $(\mathbf{x} - \mu_\mathbf{k})^T \Sigma_k^{-1} (\mathbf{x} - \mu_\mathbf{k})$, where x is the word vector for the current word, can now be solved with this technique.
Take $\mathbf{b_k} = (\mathbf{x} - \mu_\mathbf{k})$, then:

$$\underbrace{(\mathbf{x} - \mu_\mathbf{k})^T}_{\mathbf{b_k}^T} \; \Sigma_k^{-1} \; \underbrace{(\mathbf{x} - \mu_\mathbf{k})}_{\mathbf{b_k}} = \mathbf{b_k}^T \; \Sigma_k^{-1} \; \mathbf{b_k} \tag{3.7}$$

Due to the symmetry and positive definite characteristics of covariance Σ_k, we can apply the Cholesky factorization as follows:

$$\mathbf{b_k}^T \, \Sigma_k^{-1} \, \mathbf{b_k} = \textit{[by 3.6]} = \mathbf{b_k}^T (\mathbf{C_k} \mathbf{C_k}^T)^{-1} \mathbf{b_k} = \mathbf{b_k}^T (\mathbf{C_k}^{-1})^T \, \mathbf{C_k^{-1}} \mathbf{b_k} = (\mathbf{C_k}^{-1} \mathbf{b_k})^T \, (\mathbf{C_k}^{-1} \mathbf{b_k}).$$

The factorization can also be used to compute the determinant $|\mathbf{\Sigma}_k|$:

$$log(|\mathbf{\Sigma}_k|) = 2 \times \sum_{m=1}^{M} log(\mathbf{C}_{k_{m,m}}) \tag{3.8}$$

Due to this factorization, the covariance in Equation 3.3 and 3.4 can be updated by *rank one downdate* and *rank one update* respectively on C_k.
In pseudo-code, the algorithm for the *rank 1 update* is as follow [13]:
- input: A triangular Cholesky factor $C \in \mathbb{R}^{MM}$ of the covariance Σ and $z \in \mathbb{R}^M$
- output: the updated version of C by $C + zz^T$

for $m = 1,...M$ **do**

- $r = \sqrt{C_{mm}^2 + z_m^2}$
- $C_{mm} = r$
- $r^* = \frac{r}{C_{mm}}$
- $z^* = \frac{z_m}{C_{mm}}$

for $k = m+1,...,M$ **do**

- $C_{mk} = \frac{C_{mk} + z^* z_k}{r^*}$
- $z_k = r^* z_k - z^* C_{mk}$

end

end

Algorithm 3.3: Rank one update

The pseudo code above can easily be adapted to do a *rank 1 downdate*: one merely needs to replace the two red additions by subtractions.

There are other techniques that also speed up an algorithm and shorten the running time. Some of them that have been used in the GLDA case are as follow:

- **Refactor**: Short Python programs and splitting the functions into mini-functions helps a lot where each small function takes care of preferably only one task. Large functions fill up the memory and slow down the calculations. Clever coding decreases the computation time drastically specially when coding high dimensional models such as GLDA.

- **Numba package:** Using some python package such as Numba and compiling python code with @jit lets you significantly improve the speed of your code.
  To avoid compilation times each time you invoke a Python program, you can instruct Numba by @numba.jit(cache=True)[5] to write the result of the function compilation into a file-based cache. This is very effective for *rank 1 update or downdate* in Algorithm 3.3 and speed up the GLDA in our experiment 3.4 times faster.

[5] `https://numba.pydata.org/numba-doc/dev/user/jit.html`

3.3 Pointwise Mutual Information

According to Das et al [3], it is not correct to compare *perplexities* or *likelihood on a held-out test data* when comparing discrete models with continuous models. The perplexity is considered as a suitable measurement only in discrete cases such as LDA.
Also, based on the results of Chang (2009) [9], a higher likelihood of held-out documents (or lower negative log likelihood) doesn't necessarily correspond to human perception of topic coherence. They mean that a topic model which performs better on held-out likelihood may infer less semantically meaningful topics. Instead, the paper suggests to follow Newman's method (2009) [10] and choose Pointwise Mutual Information (PMI) of top- or topic words as a measurement of performance.

PMI is a function of pairs of words and the score of a topic is computed by averaging the scores of pairly co-occurrence statistics over top words. The PMI score of a model is achieved by averaging the PMI scores of all the topics. This gives all comparison of topic coherence for different models. The PMI of two top words w_i and w_j is calculated as follows:

$$PMI(w_i, w_j) = \log\left(\frac{p(w_i, w_j)}{p(w_i)p(w_j)}\right) \tag{3.9}$$

where $p(w_i, w_j)$ is the relative frequency of documents where both words w_i and w_j occur and $p(w_i)$ and $p(w_j)$ are the relative frequencies of documents where the two words appear, respectively. A model with a high PMI value is considered to have a better performance in term of topic coherence than a model with a lower one.
The corpora for the measuring of the PMI values will be a part of Storytel's database, consisting of 141,389,251 words.

In the next section we will extend GLDA to a multilingual model.

4 Multilingual GLDA

We present two new techniques of multilingual modeling that work well. First, a *joint* and *aligned* word embedding will be trained by *Fasttext* on the multilingual corpus. Then, by applying the first technique, the topic modeling on the target language will be more comprehensive and by the second one, the running time will be shortened.

Some assumptions are required in order to make cross-lingual connection between the languages practical. As has been mentioned before, in both LDA and GLDA, the documents are considered as a mixture of a number of topics. Here we go further and assume that the documents of different languages should have this property, in other words, the same *topic distribution*. It would otherwise be difficult to train two distinct documents of different languages in parallel, if for instance, one is about astronomy and other about Tinder reviews.

Additionally we assume that the words of different languages are independent of each other. This means, for instance that, the same word (but in two different languages) can be present in different topic distributions with different probabilities without any requirement of having identical topic distributions. Due to this simple yet essential assumption, the words in different languages with similar topics are aligned so that the topic distributions of different languages are aligned.

Before we apply the techniques on the GLDA, we need to make the word vectors multilingually aligned. This part is important because the same word in two different languages can have different location on the embedding space. Combining non-aligned embeddings with parameters from another language might make the covariance singular and unstable.

Facebook's Fasttext give us the opportunity to design own desirable word embedding by adjustment of the hyper-parameters even on a small corpora. It is possible to make a joint multilingual word embedding which is trained on a mixed multilingual text data, all together. In this way, we align the word embedding of different languages into the same *vector space* and then simply apply the multilingual models. Better alignment achieves when two languages are closely related to each other.
Using this kind of the word embedding makes also the embedding richer and enhances the ability of capturing the topics by the GLDA model. This is partly thanks to the sub-word information of the Fasttext.

Fasttext's *word-ngram* consider each word w as a bag of n-gram characters $\mathcal{G}_w$ and provide sharing of the information between the words in a corpus.
Let's explain it by an example to make it clear. Suppose we have the word w: *university*. First we add "<" at the beginning and ">" at the end of the word (*<university>*) to distinguish prefixes and suffixes from other character sequences. Take

the n = 3, then a set of n-grams $\mathcal{G}_w$ will be: *(<un, uni, niv, ive, ver, ers, rsi, sit, ity, ty>)* + the word itself *<university>*. Note that the sequence <sit> corresponding to the word "sit", is different from the tri-gram sit here. Each subgram g is now represented by a word representation $\mathbf{z}_g$ from a ngram dictionary [11]. Then the final word vector $\mathbf{u}_w$ for the word *university* will be sum of word representations, $\mathbf{u}_w = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g$.
The n-gram set of the corresponding Swedish one is *(<un, uni, niv, ive, ver, ers, rsi, sit, ite, tet, et>, <universitet>)*. As can be seen here, the both words share information and internal structure in the 8 first positions.

Sharing the representations across the words allows the model to learn reliable representation for the rare words and to learn rich representations from a much smaller data set. A rich embedding in its turn, supports the model to a more comprehensive and precise topic modeling.
The more related and close the languages are to each other the more cross-lingual interaction and sharing of character level information.

Now the first technique is simple since we don't involve any statistics or parameters from the source language into the targets. The fact is that the GLDA model will be trained on each language in parallel but with a *trained* joint (multilingual) word embedding.
Training a joint embedding by Fasttext on the mixed corpus, is a requirement for this technique, partly because *pre-trained* embedding from non-Fasttext are neither aligned or provided in other languages than English and partly because Fasttext provide the tool for one to train its own embedding and it can handle unseen words.

The Equation 3.2 replaces by Equation 4.1 as follows:

$$p(z_{d,i}^{(\ell)} = k | z_{\neg(d,i)}^{(\ell)}, \mathbf{V}_d^{(\ell)}, \zeta^{(\ell)}, \alpha) = (n_{d,k}^{(\ell)} + \alpha) \cdot$$
$$\cdot \frac{\Gamma[\frac{df_k^{(\ell)} + M}{2}]}{\Gamma(\frac{df^{(\ell)}}{2})\ df_k^{(\ell)\frac{M}{2}}\ \pi^{\frac{M}{2}} |\Sigma_k|^{-1(\ell)}} \left(1 + \frac{(\mathbf{x}^{(\ell)} - \mu_{\mathbf{k}}{}^{(\ell)})^T \Sigma_k^{-1(\ell)} (\mathbf{x}^{(\ell)} - \mu_{\mathbf{k}}{}^{(\ell)})}{df_k^{(\ell)}}\right)^{-\frac{df_k^{(\ell)}+M}{2}} \tag{4.1}$$

$\ell = 1, ..., L$ and L is number of languages. This technique is described in Algorithm 4.1.

The corresponding generative process of the algorithm is defined in Algorithm 4.2.

- Mix the corpus of language $l \in [1, L]$ and Create an joint word embedding
for *each language* $\ell \in [1, L]$ **do**
  for *each iteration* $i \in [1, I]$ **do**
    for *each document* $d \in [1, D^{(\ell)}]$ **do**
      for *each word* $n \in [1, N_d^{(\ell)}]$ **do**
        - $n^{(\ell)}[d, k] = n^{(\ell)}[d, k] - 1$
        - Update topic parameters ${\mu_{\mathbf{k}}}^{(\ell)}$ and ${\mathbf{\Sigma_k}}^{(\ell)}$ by rank one downdate
        for *each topic* $k \in [1, K]$ **do**
          -Calculate probability of topic k, $p(z_{d,n}^{(\ell)} = k | z_{\neg(d,i)}^{(\ell)}, \mathbf{V}_d^{(\ell)}, \zeta^{(\ell)}, \alpha)$ by Equation 4.1
        end
        - Normalize the probability vector $\mathbf{p}_{d,n}^{(\ell)} = [p_1, p_2, ..., p_k]$ to sum of one.
        - Sample a topic k from $\mathbf{p}^{(\ell)}$ and assign it to the current word
        - $n^{(\ell)}[d, k] = n^{(\ell)}[d, k] + 1$
        - Update topic parameters: ${\mu_{\mathbf{k}}}^{(\ell)}$ and ${\mathbf{\Sigma_k}}^{(\ell)}$ by rank 1 update
      end
    end
  end
end

Algorithm 4.1: Simple training algorithm for MGLDA with Gibbs sampling

for *each language* $l \in [1, L]$ **do**
  for *each topic* $k \in [1, K]$ **do**
    - Draw prior topic covariance $\Sigma_k^{(l)} \sim W^{-1}(\Psi^{(l)}, \nu^{(l)})$
    - Draw topic mean $\mu_k^{(l)} \sim N(\mu^{(l)}, \frac{1}{k}\Sigma_k^{(l)})$
  end
end
for *each language* $l \in [1, L]$ **do**
  for *each document* $d \in [1, D^{(l)}]$ **do**
    -Draw topic distribution $\theta_d \sim Dir_k(\alpha)$;
    for *each word* $n \in [1, N_d^{(l)}]$ **do**
      -Draw a topic $z_n^{(l)} \sim Cat(\theta_d)$
      -Draw a word $v_{d,n}^{(l)} \sim N(\mu_{z_n}^{(l)}, \Sigma_{z_n}^{(l)})$
    end
  end
end

Algorithm 4.2: Generative process of MGLDA

GLDA with the multilingual word embedding yields more comprehensive and precise topic prediction on the target corpus than with the embedding that is trained only on the target language. This method is especially favorable when we don't have access to a rich text data of the target language, but instead have quite enough in the source language.

The algorithm indeed, has not really much to do with GLDA but is a way to utilize the way that Fasttext works in order to train an embedding that learns better by using a larger database even if the database expands only with documents from another languages.

In order to gain maximum benefit from the sub-word information across the languages (and consequently more comprehensive topic prediction) we think that the languages need to be closely related.

It should be noted that Facebook has published *pre-trained aligned* word vectors for 44 languages [6] which would be a natural first choice to use here. However it turns out that one can create, by the first technique of MGLDA, an alignment (between two languages) that seems to work better than the one the *pre-trained aligned* provide. Unfortunately the topic classification, using these *pre-trained* word embeddings (tested on bilingual corpus from Amazon reviews), are far from comprehensive in comparison with the results from this technique.

Next we move to our second multilingual technique. Here information from the source language will be borrowed to be used to speed up the inference on the target languages. More precisely, the topic parameters $\mu_k^{(1)}$ and $\Sigma_k^{(1)}$ which are trained on the source language will be set as initial values for $\mu_k^{(\ell)}$ and $\Sigma_k^{(\ell)}$ in the training for the target languages, where $\ell \in [2, L]$.
This procedure makes it faster for Gibbs sampling to converge in corpus of the target languages and spares us the running time and number of iterations.
Since parameters from the source have been used in the targets one can call the algorithm, approximate MGLDA. The method is described step by step in Algorithm 4.3.

This technique is a developed case of the previous multilingual algorithm (simple MGLDA) which means that one can achieve two goals, first more comprehensive topic prediction as the previous technique and then shorter computational time on corpus of target languages.

[6] `https://fasttext.cc/docs/en/aligned-vectors.html`

- Mix the corpus of language $l \in [1, L]$ and train an joint word embedding
for *source language (1)* **do**
  for *each iteration of the source language* $i \in [i, I^{(1)}]$ **do**
    for *each document* $d \in [1, D^{(1)}]$ *of the source language* **do**
      for *each word* $n \in [1, N_d^{(1)}]$ **do**
        - $n^{(1)}[d, k] = n^{(1)}[d, k] - 1$
        - Update topic parameters ${\mu_{\mathbf{k}}}^{(1)}$ and ${\mathbf{\Sigma_k}}^{(1)}$ by rank one downdate
        for *each topic* $k \in [1, K]$ **do**
          -Calculate probability of topic k, $p(z_{d,n}^{(1)} = k | z_{\neg(d,i)}^{(1)}, \mathbf{V}_d^{(1)}, \zeta^{(1)}, \alpha)$ by Equation 3.2
        end
        - Normalize the probability vector $\mathbf{p}_{d,n}^{(1)} = [p_1, p_2, ..., p_k]$ to sum of one.
        - Sample a topic k from $\mathbf{p}$ and assign it to the current word
        - $n^{(1)}[d, k] = n^{(1)}[d, k] + 1$
        - Update topic parameters: ${\mu_{\mathbf{k}}}^{(1)}$ and ${\mathbf{\Sigma_k}}^{(1)}$ by rank 1 update
      end
    end
  end
  - spare ${\mu_{\mathbf{k}}}^{I,(1)}, {\mathbf{\Sigma_k}}^{I,(1)}$ from the last iteration
end
for *each language of the target languages* $\ell \in [2, L]$ **do**
  for *each topic* $k \in [1, K]$ **do**
    - Initialize $\mu_k^{0,(\ell)} = \mu_k^{I,(1)}$, $\Sigma_k^{0,(\ell)} = \Sigma_k^{I,(1)}$
  end
  for *each iteration of the target language* $i \in [1, I^{(\ell)}]$ **do**
    for *each document* $d \in [1, D^{(\ell)}]$ *of the target language* **do**
      for *each word* $n \in [1, N_d^{(\ell)}]$ **do**
        - $n^{(\ell)}[d, k] = n^{(\ell)}[d, k] - 1$
        - Update topic parameters ${\mu_{\mathbf{k}}}^{(\ell)}$ and ${\mathbf{\Sigma_k}}^{(\ell)}$ by rank one downdate
        for *each topic* $k \in [1, K]$ **do**
          -Calculate probability of topic k, $p(z_{d,n}^{(\ell)} = k | z_{\neg(d,i)}^{(\ell)}, \mathbf{V}_d^{(\ell)}, \zeta^{(\ell)}, \alpha)$ by Equation 4.1
        end
        - Normalize the probability vector $\mathbf{p}_{d,n}^{(\ell)} = [p_1, p_2, ..., p_k]$ to sum of one.
        - Sample a topic k from $\mathbf{p}^{(\ell)}$ and assign it to the current word
        - $n^{(\ell)}[d, k] = n^{(\ell)}[d, k] + 1$
        - Update topic parameters: ${\mu_{\mathbf{k}}}^{(\ell)}$ and ${\mathbf{\Sigma_k}}^{(\ell)}$ by rank 1 update
      end
    end
  end
end

Algorithm 4.3: Approximate training algorithm for MGLDA

5 Results and Experiments

In this section, the models that have been described earlier such as LDA, GLDA with *pre-trained*, GLDA with *trained* embedding, simple MGLDA and approximative MGLDA will be applied and tested on two different data sets. These data sets are imported from two different sources: Amazon website and Storytel's database.

First a file named *all-sentiment-shuffled.txt*, containing around 12,000 reviews of Amazon products, has been downloaded[7] in order to test LDA and both cases of GLDA. Then, for testing multilingual models, 20% of them have been translated into Swedish by Google Translate.

These reviews are mostly about 6 product types: camera, DVD, music, software, book and other products such as health product used in the paper by Blitzer et al., (2007) [14]. The data has been formatted so that there is one review per line. For simplicity we assume that the true topics of the reviews are these 6 types, but other semantically related top words (that have been captured by the models) will be presented as well. In some tables a descriptive word (marked with brown text color) about the topic that top words represent, has been added to the top of columns.

The rest of the data is from Storytel's database. In order to test LDA and both GLDA cases, a collection of ten documents consisting of 204,159 words (after pre-processing), has been selected from the database. The selection is randomly which means that the documents can contain completely different and unknown topics. This makes, in its turn, the classification an unsupervised task also for us as human.

In the Bilingual case, both MGLDA models have been tested on a novel of Beate Grimsrud called, *"A Fool Free"* with corresponding Swedish edition: *"En Dåre Fri"*. It's about a girl who lives alone and hears voices in her head. She is periodically admitted to the psychiatric ward. The Swedish one consists of 54,025 words after pre-processing.

[7] `https://svn.spraakdata.gu.se/repos/richard/pub/statnlp2015_web/assignment1.html`

5.1 Results from LDA and GLDA models

The output or top words (in ascending order) of the monolingual models such as LDA by algorithm 2.2, GLDA by algorithm 3.2 both with *pre-trained* and *trained* word embedding are shown in the coming tables: 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6. The first three tables correspond to the output of the models applied on Amazon data and the latter three on the Storytel's collection of 10 documents.

Tables 5.1, 5.2 and 5.3—LDA, GLDA with *pre-trained*, and GLDA with *trained* embedding respectively— have been trained on 10% of the reviews. In both GLDA cases (*pre-trained* and *trained*), the number of Gibbs iterations and number of topics K have been set to 5 and 50 respectively but in LDA to 50 and 12 because of poor performance if a lower respectively a higher numbers have been chosen. The word embedding in Table 5.2 is from Fasttext's *pre-trained*: wiki.news.300d.1M.vec and in Table 5.3 is trained on the entire Amazon reviews with the dimension M equal to 150. The GLDA in latter case is then run on 10% of the reviews afterwards.
In LDA (Table 5.1), despite the high number of iterations, some of the top words in respective columns are not semantically related to others. On the other hand, in both GLDA cases almost all the true topics have been successfully classified, and the models captured some other topics (semantically related top words) as well.
These new topics in Table 5.2, seem to be family member words, colors, numbers, positive adjectives, names and Emoji words which are different from the new ones in Table 5.3.
That these "extra" topics seem to be so coherent is an effect of the fact that the embedding itself captures the semantics of the language and the fact that GLDA identifies topics with multivariate Gaussian distributions and that the word vectors close to that mode will necessarily be semantically close.
One thing to consider is that these extra topics can slightly variate when training several times but variation in capturing the true topics is low and minimal.

The total running time, in *trained* case, was 68 minutes of which 2.5 minutes for creating the word embedding. This time is 4.4 times shorter than the running time with a *pre-trained* word embedding.

Tables 5.4, 5.5 and 5.6—LDA, GLDA with *pre-trained*, and GLDA with *trained* embedding respectively— have been trained on a collection of the 10 documents from Storytel.
The number of topics in LDA and both GLDA cases are set to 50 but the number of Gibbs iterations are set to 30, 5 and 10 in respective cases. The GLDA model in *trained* case, required more iterations (10 iterations) than the *pre-trained* one because it took longer time for negative log likelihood to keep an ascending trend.
The word embedding in Table 5.5 is Fasttext's *pre-trained* one, named: wiki.en.vec and in Table 5.6, it has been trained on these 10 documents before applying the GLDA on them.

It seems that the LDA (in table 5.4) captured some topics in areas such as: medicine, music, traffic (twice), school related words but not semantically precise.
The captured ones in GLDA with *pre-trained* embedding (in table 5.5) are in areas such as: medicine, body parts, music, riding, negative feelings, top names within Islams history (shiite and sunni), building parts, accidents, service system and process related words. Most topics in *trained* case, in Table 5.6, are the same as in *pre-trained* one. It can been seen that the semantic relation between the top words in both GLDA cases are higher than the one in LDA model in table 5.4.
It might be good to mention that there are ,in GLDA cases, some other semantically related top words of other topics as well but we chose not to present them here.

Since the true topics in this collection are unknown for us, the PMI values have been used as a measurement of the performance. The corresponding PMI value for each topic are marked with blue text color under the corresponding column. The average PMI of the top words in LDA and respective GLDA models are 10.09, 11.53 and 12.08 which indicates that the GLDA has a better performance and topic coherence than the LDA.

Table 5.1: Top words in ascending order of 8 topics from the LDA model by Algorithm 2.2. The model have been trained on 10% of the Amazon reviews.

digital	war	people	zoom	really	etc	better	bought
well	scenes	go	look	characters	using	guitar	need
battery	great	let	back	written	makes	wa	used
canon	acting	wa	product	made	fact	much	computer
used	ever	think	problem	know	series	spanish	days
easy	two	enough	hair	much	several	mm	first
product	best	buy	light	books	information	version	price
could	make	great	water	author	little	band	last
bought	even	music	really	say	words	album	new
software	love	good	right	book	true	songs	also
pictures	music	best	also	story	guess	cd	work
camera	dvd	also	use	read	find	lens	version

Table 5.2: Top words in ascending order of 12 topics from the Gaussian LDA model with Fasttext's *pre-trained* word embedding named: wiki.news.300d.1M.vec. The model is run on 10% of Amazon reviews.

camera	DVD	music	software	book	family member
camera	cd	songs	software	movie	father
cameras	dvd	album	version	book	mother
lens	gman	music	concept	film	wife
lenses	divx	song	system	story	man
photography	zune	albums	aspect	movies	girl
photographs	bluray	tunes	approach	films	friend
pictures	cds	singer	thing	novel	boy
photographer	panasonic	lyrics	program	books	daughter
viewfinder	cdrom	recordings	idea	sequel	brother
photos	windoze	musician	problem	novels	son
camcorder	flac	band	picture	screenplay	woman
zoom	winxp	melodies	data	documentary	person
color	numbers	positive adjectives	trade	names	emoji
brown	two	wonderful	bought	bellamy	suks
water	three	good	buy	phillip	okk
metal	several	nice	purchased	loretta	OMG
yellow	four	fantastic	sold	damien	Hahahahahah
skin	five	great	sell	eleanor	pplz
plastic	many	amazing	purchase	ronald	jeezus
white	first	terrific	buying	rhys	wazzup
blue	six	fabulous	selling	stacey	wuts
hair	other	terrible	owned	crowe	peepz
glass	seven	brilliant	built	nicholas	lolololololol
black	numerous	interesting	rented	dwayne	mmmmmmmmm
red	previous	lovely	shipped	cassidy	yea

Table 5.3: Top words in ascending order of 12 topics from the Gaussian LDA model with *trained* word embedding. The embedding is trained on the entire Amazon reviews. The GLDA model is then run on 10% of the same data afterwards.

camera	DVD	music	software	book	quantity
camera	dvd	album	software	book	small
lens	movie	songs	computer	read	size
canon	watch	song	program	books	smallpox
cameras	movies	music	install	reading	smalltown
lenses	tv	cd	softwares	author	smallish
nikon	watching	tracks	windows	written	larger
digital	series	albums	installer	novel	smaller
pictures	show	lyrics	vista	chapters	unit
flash	films	listening	system	publishers	large
battery	video	band	preinstalled	textbooks	fit
slingshot	fullscreen	instrumental	pc	wading	double
zoom	netflix	tunes	xp	story	diameter
spending	quality	problem	version	sound	product
money	quality	problem	version	sound	use
waste	bit	problems	original	listen	product
worth	good	fix	versions	sounds	used
spend	overall	solved	released	hear	works
spending	quite	conection	rereleased	recordings	bought
save	far	issue	latest	listening	using
buy	pretty	unsolved	editions	homerecording	work
hardearned	high	seem	previous	soundworld	products
wasting	poor	tried	rerecorded	band	packages
pay	excellent	troubleshooting	upgraded	bluesrock	purchases
dollars	great	complains	updated	live	inexpensive
refunded	compared	malfunction	remastered	rock	serum

Table 5.4: Top words in ascending order of the LDA model and their corresponding PMI values by blue color. The LDA is run on the collection of 10 documents of the Storytel's database.

somewhere	scene	journeys	traffic	matches
hospital	yeti	holes	avoid	librarian
never	song	scale	trainer	tweynine
nurse	squawker	notes	training	jokes
morning	ha	chapter	examiner	lake
major	cat	note	need	papers
bed	mabel	enchaed	you're	values
john	mayor	play	must	lad's
though	fit	guide	use	swank
soon	wiggles	flutes	you'll	sports
us	susie	style	road	paracetamol
say	halloweena	american	riding	chess
seemed	audience	native	motorcycle	huey
tom	johnnie	palying	ride	esther
said	bessie	flute	test	arthur
8.94	12.23	10.54	9.30	11.92
coll	him	traffic	technology	time
man	dayroom	braking	chapter	knowledge
iraq	behaviour	chapter	week	business
truck	illness	time	hours	figure
plane	grounds	vehicle	you'll	change
anwar	triplets	information	work	support
black	this	hazards	lesson	table
us	carrier	vehicles	you're	known
imjiw	'come	use	pmpm	root
khalid	garden	position	step	record
looked	began	machine	study	servic
preside	peter	riding	learning	incide
caliph	doors	see	course	process
men	title	speed	elearning	manageme
stanley	peter	road	online	problem
10.54	9.72	9.60	9.22	8.92

Table 5.5: Top words in ascending order of the GLDA with Fasttext's *pre-trained* embedding named: wiki.en.vec and their corresponding PMI values. The GLDA is run on the collection of 10 documents of the Storytel's database

medical	eyes	flute	riders	telling
obstetrics	eyelids	flutes	rider	embarrassed
pediatrics	ears	percussively	men	worried
pediatric	forehead	percussive	rode	annoyed
gynecology	lips	fingering	cyclists	afraid
pediatrician	eyebrows	flutemaker	motorcyclists	ashamed
surgeons	fingers	orchestration	women	scared
surgeon	cheeks	instruments	cyclist	disgusted
cardiology	fingernails	guitar	motorcyclist	knowing
gynecologist	eyelid	horns	bike	terrified
physician	hair	brass	ride	amused
pediatricians	noses	pentatonic	bikes	angry
doctors	eyeballs	bass	motorcycle	disappointed
physicians	nostrils	instrument	motorcycling	frightened
cardiologists	skin	drums	dismounting	horrified
urologist	fingertips	flutemakers	bicycles	surprised
hospital	nose	symphonic	sprinting	thrilled
clinic	eye	octaves	horses	infuriating
urology	cheekbones	keyboard	motorbike	frustrated
orthopedic	thighs	fluting	downhill	astonished
11.78	9.99	12.02	10.83	10.11
khalid	room	incident	service	process
bakr	hallway	incidents	services	processes
caliph	upstairs	accidents	providers	procedures
muhammad	door	accident	provider	procedure
caliphs	stairwell	incident	selfservice	interprocess
abdul	downstairs	deaths	hourly	processing
imam	bathroom	altercation	customer	mechanisms
caliphate	rooms	happened	customers	processed
abu	bedrooms	fatality	staffing	entails
shiite	bedroom	collision	provision	workflows
hussein	floor	kidnappings	timetables	workflow
sunnis	balcony	occurred	operated	methods
khalids	doors	crash	uniformed	mechanism
fakhr	doorway	repercussions	agencies	method
anwar	stairs	horrific	firstline	automating
hassan	staircase	disaster	delivery	prioritization
ahmed	foyer	aftermath	operate	rationalization
sunni	basement	allegations	serving	application
ali	cloakroom	bystanders	ambulance	involves
gassan	cupboard	confrontation	personnel	system
17.09	9.95	10.95	11.09	11.45

Table 5.6: Top words in ascending order of the GLDA model with *trained* embedding, created from the collection of 10 documents of the Storytel's database, and their corresponding PMI values. The GLDA is then run on the same data set.

inexcusable	road	enchants	riding	vehicles
expendable	carriageway	enchanting	multitasking	vehicle
philosophypsychology	carriageways	flutemaker	physiological	overtake
pheromones	roadside	flutemakers	minimising	overtaken
chorioamnionitis	osmpsl	enchanted	riders	motorcyclists
scissors	crossroads	styles	vulnerability	checklist
anthropology	roadworks	quickstart	riders	driver
inexcusable	markings	chaplin	risks	follows
gastrointestinal	oneway	kickstart	attitudes	crossroads
anesthetic	slowspeed	flute	rider	motorcyclist
philadelphia	roundabouts	penchant	vulnerable	twoway
brooklyn	irregularities	antiamerican	affects	drivers
hematocrits	slope	style	slowspeed	obstacles
hyperkinetic	multilane	flutecrafting	selfassessment	cyclists
hertfordshire	curve	dilute	downhill	parked
13.67	12.57	13.70	11.68	11.27
imjiw	slowspeed	problem	service	process
furthermore	motorcycle	management	restoration	processes
sunni	speed	managements	services	processing
jihad	machines	managers	workflow	organization
kurdish	roadworthiness	csfs	desktop	procedures
muslims	motorcycles	manager	outages	csfs
syrian	motorways	problems	supplier	organizational
infidels	motorcyclist	incident	disruptions	reorganization
soldiers	ride	kpis	reduction	proceed
mujahideen	roadworks	implementing	hierarchical	organizations
islamic	speeding	matrix	outlook	procedure
syria	osmpsl	occurrences	suppliers	make
beers	speeds	inputs	customerfacing	workflow
squish	motorcyclists	implement	postimplementation	highlevel
islam	controlling	activities	applications	measurements
12.57	12.17	10.77	11.66	10.81

5.2 Results from Multilingual GLDA models

In the Bilingual case, both MGLDA models i.e. simple MGLDA and approximate MGLDA, have been tasted on bilingual texts from both Amazon website and Storytel's database. The tables 5.7 and 5.8 regards simple MGLDA and tables 5.9 and 5.10 approximate MGLDA model.

20% of the reviews have been translated into English so the both Swedish and English texts share the same *topic distribution*. The Swedish data then divided into two parts with distinct number of documents and vocabulary size. A joint word embedding is trained on the entire English version together with the first half of the Swedish corpus. Then the embedding is used for topic modeling on the second half of the Swedish data.
This result is compared with the monolingual embedding case (trained only on the first half of the Swedish corpus) in Table 5.7 where the upper part of the table consists of top topic words by Algorithm 4.1 and the lower part of top words by the monolingual Swedish word vector.
For randomness the same seed function has been used and the number of Gibbs iterations and topics are 5 and 50 respectively in both cases.
It seems that the topics Camera (Kamera on Swedish), software (programvara on Swedish) and book (bok) are captured by both. The monolingual case has almost missed the topic music which is captured by bilingual one. The missing occurs even when trying with different numbers of topics. The occurrence of English words (red colored) is rare despite the fact that the majority of words in the embedding vectors are in English.

Algorithm 4.1 have also been applied on the bilingual novels *"A Fool Free"* and *"En Dåre Fri"*. The results are shown in Table 5.8 where the upper part is for simple MGLDA and the lower part for monolingual model. The procedure is the same as the previous experiment i.e. first a joint embedding is trained on both Swedish and English novels together, then the model have been trained on the Swedish one. At the same time a monolingual embedding is trained only on Swedish for monolingual GLDA.
The number of iterations, topics and the dimension of the word embedding are 10, 4 respectively 50 in both cases.
In the upper part ot the table, topics 1 and 3 seem to represent the same topic: care. Topic 2 says nothing specific and topic 4 is about forced psychiatric treatment. In the lower part, topics 1, 2, and 4 seem to represent the same topic i.e. care and topic 3 is not represented. The simple multilingual model has captured one more topic than the corresponding monolingual one.

In order to test Algorithm 4.3 (approximate MGLDA), also two experiments on the same bilingual data sets: Amazon reviews and Storytel's novels have been carried out.

In Amazon case, the entire data set in English is taken as the source language

and 10% of the same data in Swedish is used as target language. A joint word embedding is trained on the mixed bilingual corpus. The GLDA model is first trained on the 10% of the source data, which corresponds to the Swedish one, with 5 iterations. Then the topic parameters of the last iteration are used as initial values for the parameters in the target corpus. The GLDA is then run on the target data only up to three iterations. The outcomes of the third iteration of the target language by approximate multilingual model and monolingual one are shown in upper respectively lower part of Table 5.9.

It can been seen from the table that the multilingual model has carried out a better topic prediction than the monolingual case since the latter one missed totally the topic: software and is not as precise as the multilingual one in DVD and book. The words from the source language (English), marked by red color, are sometimes found in the upper one since we use a joint word embedding.
Note that because of the time consuming we don't train the GLDA on entire source data.

Table 5.10 regards the second experiment of approximate MGLDA (on the novels). The number of topics and the dimension of the word embedding are 4 and 50 respectively, but number of iterations is 5 which is half the number of iterations in the monolingual case (the lower part of the Table 5.8). The words from the source language i.e. English, is marked by red color.
Topic 3 and 4 seem to represent the same topic: care. Topic 2 is almost about forced psychiatric treatment. Topic 1 dominates by names and English words. Again despite the lower number of iterations, the model captured one more topic than the monolingual case. In addition, the model spared us the running time and several number of iterations.

Table 5.7: The upper table consists of top words by Algorithm 4.1 (simple MGLDA) and the lower table by the monolingual GLDA. For both part, the same data i.e. 10% of the Amazon data have been used.

Camera	DVD	Music	software	book	product
kamera	film	album	programvara	bok	produkt
kameran	filmskapare	albums	programmet	fascinerande	betaprodukt
bilder	skräckfilm	band	programvaran	omtvistad	produktens
kameraväska	filmerna	albumen	program	författarens	produktsidan
inomhusbilder	skådespelarna	song	installera	tvivelaktiga	produktlinje
bilderna	budskap	cd	installerar	kvardröjande	produkter
digitalkamera	sköljer	songs	installerade	filosofer	företagsprodukt
utomhus	skådespeleriet	rock	påskynda	bokhylla	produkten
videokameran	tacksam	debutalbum	programmering	synpunkter	produktbilder
bilden	manuset	soloalbum	installer	letade	officeprodukt
suddiga	inspektör	studioalbum	leverantörerna	chockerande	kontorsprodukter
bildskärm	inspirerande	music	automatiska	öde	adobeprodukter

Camera	DVD	Music	software	book	product
kamera	filmen	cd	program	bok	produkt
kameran	film	debutalbum	programmet	boken	produkter
kameraväska	filmer	lyssnade	programvara	bokhylla	produkten
digitalkamera	filmerna	gorillaz	programvaran	trumpetade	produktion
kameror	filmskapare	träffat	programmering	filosofier	produktionen
videokameran	dialogen	jason	ritprogram	läsningen	returnera
digitalkameror	dialog	demon	pro	tips	pro
videokamera	jet	bonusspår	acrobat	författare	returneras
videokameror	ronald	ångra	quad	boot	produceras
lcdskärmen	hyllning	lynnat	bärbar	leonardo	salicylsyra
lenma	skådespeleriet	debut	penna	splittingly	rakhyvel
batteri	skådespelarna	grace	laptop	böcker	projekt

Table 5.8: The upper table consists of top words by Algorithm 4.1 (simple MGLDA) and the lower one by monolingual GLDA. The data for both cases is the Novel from Stroytel, called *"En Dåre Fri"*.

topic 1	topic 2	topic 3	topic 4
ursäkt	säger	ursäkt	emellan
jesus	språk	jesus	sjukhusets
sjuksköterskan	väldigt	sjuksköterskan	centre
sjukdomen	då	råd	polischefen
mjuk	säg	mjuk	hastigt
kostym	schemat	ide	polisen
konstnär	någon	sjukdom	sjuksköterskan
råd	bra	sjukhuset	ursäkt
ide	ni	ungefär	vård
ihåg	lyder	lugnande	camera
frysa	lyda	skydd	desperate
här	säg	ihåg	ungefär
lugnande	sprutan	förhandlingarna	tvångsvård
nyfiken	uppmärksamhet	levande	dig
klunkar	läkaren	kostym	ytterligare
sjukhuset	sätt	inlagd	samtal
skydd	piper	konstnär	engelska
inlagd	följa	gräva	ide
levande	dunker	frysa	lunatic

topic 1	topic 2	topic 3	topic 4
ytterligare	patienten	vill	patient
patient	ytterligare	bara	patienten
ursäkt	ursäkt	vilja	ytterligare
patienten	patient	elsa	ursäkt
emellan	honung	hon	patienter
planerat	planerat	vila	emellan
samtalet	samtalet	gömt	samtalet
druckit	inlagd	värd	honung
honung	emellan	planerat	hon
patienter	skötare	permission	planerad
jobbet	producenten	ville	druckit
inlagd	patienter	tillstånd	jobbet
hon	engelska	frivilligt	pjäs
elsa	munkjacka	jämt	uppsala
engelska	jesus	gräva	jesus
jesus	oss	lyfta	elsa
munkjacka	druckit	uppmärksamhet	engelska
skötare	pjäs	smärtan	oss
ihåg	gräva	glömma	inlagd

Table 5.9: The upper table consists of top topic words by Algorithm 4.3 (approximate MGLDA) and the lower one of top words by the monolingual Swedish word vector. The data for both cases is 10% of the Amazon data.

camera	DVD	music	software	book
kamera	film	cd	datorn	bok
kameran	filmskapare	album	dator	bokhylla
kameraväska	filmen	soloalbum	datorer	djup
digitalkamera	dialogen	studioalbum	ner	författarens
videokameran	skådespeleriet	sköna	installerar	kunskap
digitalkameror	filmerna	debutalbum	installera	boken
kameror	karaktär	förgyllda	installerade	kvardröjande
videokamera	sådespeleri	begåvade	installerat	författaren
skärpa	story	skratta	användaren	begrepp
utomhus	films	sköljer	installer	kapitel
ljusförhållanden	skådespelarna	lyssnat	bländaren	bokstavligen
blixt	skräckfilm	bonusspår	påskynda	filosofer
bilder	intetsägande	utbildade	ladda	filosofier
inomhusbilder	centrerad	skrika	kabeln	synpunkter
videokameror	filmer	spåra	användarvänlighet	författarna
oskärpa	vilde	smörjer	användare	djupare
värme	utbildade	albumen	uppdateringar	omtvistad
objektet	kommande	spåren	kopiera	skandalen
blixten	skurkar	vuxna	nämne	ideer
suddiga	manuset	kommersiella	användarvänligt	kurser

camera	DVD	music	software	book
bilder	film	album		boken
bilderna	filmen	albumet		bokhylla
bilden	filmer	albumen		squash
bild	skräckfilm	soloalbum		bok
inomhusbilder	filmerna	debutalbum		hoppades
spökbilder	filmbranschen	exil		jordbrukarens
suddig	oscar	huvudgatan		livsmedelsbutiker
suddiga	lecter	studioalbum		kunskap
bryn	filmskapare	rolling		universum
bilda	sotl	bandets		indelad
kortet	dialogen	bob		ironweed
bildskärm	adam	bonusspår		romantik
vidöppen	spökbilder	solon		bänk
nyansen	dream	låtarna		lärd
byggnader	skräck	spåret		indeed
pixlar	sutherland	pågår		varken
upplösningen	tantillizing	avgång		läsningen
bildkvalitet	eliminerar	spåren		intelligent
sundet	nevad	marley		kapitel
bil	burton	rankad		budskap

Table 5.10: This table consists of top words by approximate MGLDA model corresponding to algorithm 4.3. The text data is the novel from Storytel, called *"En dåre Fri.*

topic 1	topic 2	topic 3	topic 4
marit	sjukhusets	ursäkt	ursäkt
marilyn	emellan	jesus	jesus
mats	ursäkt	mjuk	sjuksköterskan
cabin	sjuksköterskan	sjuksköterskan	sjukdomen
cats	hastigt	råd	kostym
unit	ungefär	ide	mjuk
desperate	ide	skydd	klunkar
handle	centre	sjukdom	ihåg
fire	engelska	ihåg	här
midst	vård	ungefär	frysa
armband	camera	lugnande	ide
union	jesus	förhandlingarna	råd
halfway	tvångsvård	kostym	konstnär
tray	jobbet	klunkar	lugnande
suit	journalen	sjukhuset	nyfiken
hardly	munkjacka	inlagd	skydd
mad	polischefen	levande	gå
holiday	samtal	gräva	levande
shirt	samtalet	munkjacka	inlagd
lost	koncentrera	frysa	ungefär

6 Conclusion and Future Work

In this thesis, we have studied the GLDA model in detail and compared it to the basic and simple model, the traditional LDA. We have shown that the GLDA, more than LDA, is an intelligent and practical algorithm in capturing the latent topics with semantic relationship between the top words.

This was exhibited and tested on several small and medium sized corpus, some from Amazon Reviews and others from Storytel's database. In both cases the GLDA made a better job than LDA. This was possible thanks to the structure of the GLDA and the word embedding which can be downloaded or created by Facebook's function *Fasttext*. Word vectors are becoming more and more important tools in unsupervised learning problems and natural language processing. We can train them in different dimensions and in different languages and also produce a joint multilingual word embedding by training on a mixed multilingual corpora.

We have shown that our multilingual techniques makes the GLDA model significantly better in catching the topics of data from the target languages and shorten the running time. These multilingual models are uncomplicated methods and can easily be applied on different corpus or corpora without drawing any extreme assumptions about equality in the number of documents, vocabulary or sample size of the text in different languages. It is possible that there can be other methods as well that can work as multilingual techniques for the GLDA but these will be left as future works and analysis.

As a future work it can also be interesting to expand the GLDA algorithm with topics modeled as a hidden Markov chain where the order of words matters. In this case the *bag of word* assumption will be violated.

Bibliography

[1] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 746–751, Atlanta, Georgia, June. Association for Computational Linguistics.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022, March

[3] Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for Topic Models with Word Embeddings. In Proceedings of the Meeting of the Association for Computational Linguistics (ACL). ACL, 795–804.

[4] T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. Proceedings of the National Academy of Sciences, 101:5228–5235, April.

[5] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems (pp. 3111–3119)

[6] Michael U Gutmann and Aapo Hyv¨arinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. The Journal of Machine Learning Research, 13:307–361, 2012.

[7] Jeffrey Pennington, Richard Socher, Christopher D. Manning. Glove: Global Vectors for Word Representation. Published in EMNLP 2014 DOI:10.3115/v1/D14-1162

[8] Yoshua Bengio. 2009. Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2(1), pp.1-127

[9] Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In Neural Information Processing Systems.

[10] David Newman, Sarvnaz Karimi, and Lawrence Cavedon. 2009. External evaluation of topic models. pages 11–18, December

[11] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T. (2017). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5, 135-146

[12] Tong Z, Zhang H (2016) A Text mining research based on LDA topic modelling. In: International Conference on Computer Science, Engineering and Information Technology, pp 201–210

[13] Stewart, G. W. (1998). Basic decompositions. Philadelphia: Soc. for Industrial and Applied Mathematics. ISBN 0-89871-414-1.

[14] Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In Proceedings of Association for Computational Linguistics (ACL'07).