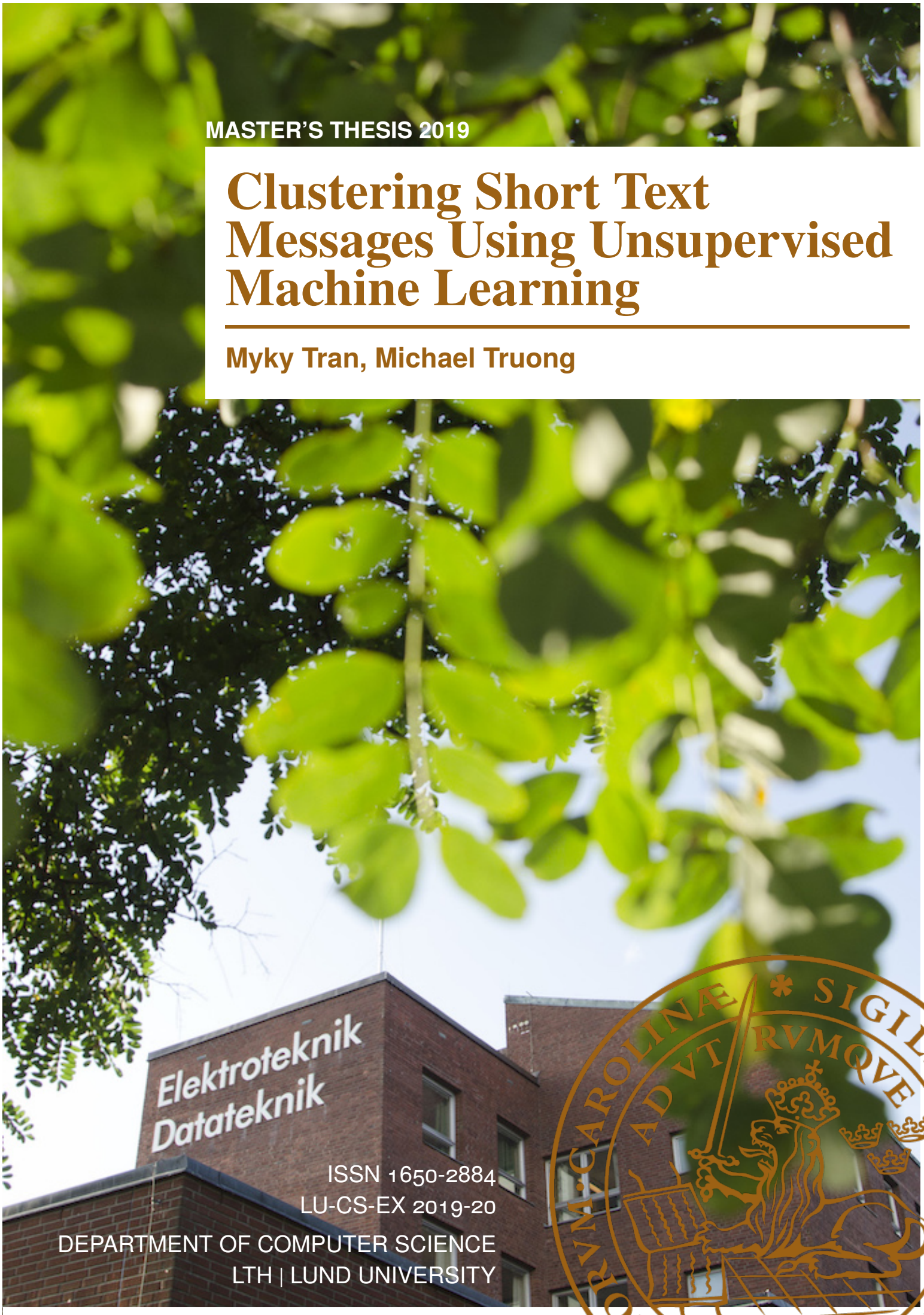


MASTER'S THESIS 2019

Clustering Short Text Messages Using Unsupervised Machine Learning

Myky Tran, Michael Truong

ISSN 1650-2884
LU-CS-EX 2019-20

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2019-20

Clustering Short Text Messages Using Unsupervised Machine Learning

Myky Tran, Michael Truong

Clustering Short Text Messages Using Unsupervised Machine Learning

(A LaTeX class)

Myky Tran
tna13mtr@student.lu.se

Michael Truong
tfy14mtr@student.lu.se

July 5, 2019

Master's thesis work carried out at Sinch AB.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Lars Novak, lars.novak@sinch.com
Jianhua Cao, jianhua.cao@sinch.com

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Today, large amounts of textual information can be found everywhere. If you have used digital platforms for reading books you have probably noticed there are algorithms recommending similar books you might like. Some recommendations are based on other reader's patterns and some are based on similar content. One class of unsupervised machine learning algorithms used for semantic analysis are called topic models. These algorithms are widely used in the industry for long texts, but in the past years, there has been an increasing interest towards analyzing short texts on digital platforms, such as SMS and social media. This is due to high interest in analyzing trending topics. Doing this presents problems in data sparsity caused by the short documents.

Topic models are statistical models that cluster words based on their co-occurrences in the documents. This results in word clusters which humans can interpret as topics, such as "weather" or "politics". On top of this, each document can be given a low dimensional representation used for clustering or labeling to allow for classification of unseen text messages.

In this thesis, we have chosen latent Dirichlet allocation and biterm topic model as our topic models. To infer the latent variables of these models, we used Gibbs sampling. In total, 15 unique topics are found, though more and better topics can be found with an improved model.

Keywords: MSc, unsupervised, machine learning, topic model, latent Dirichlet allocation, biterm topic model

Acknowledgements

We would like to thank our supervisor Pierre Nugues at Department of Computer Science for his continuous guidance and advice on our thesis work.

Further, we would like to thank our supervisors Lars Novak and Jianhua Cao at Sinch for providing us with the opportunity of doing our thesis at Sinch, as well as for the invaluable feedback and guidance along the way.

On top of that, we would like to thank our colleagues at Sinch who supported us with evaluating the results and Magnus Wiktorsson at Mathematical Statistics for helping us understanding mathematical derivations.

Finally, we would like to thank our family and friends for their continuous support.

Contents

Chapter 1
Introduction

1.1 Background

Today, more and more industries are moving towards data-driven solutions to keep up with the rapid changes in society. The telecommunication industry is not an exception. Due to the digital revolution, the way we humans interact with each other has completely changed.

In the past few years, there has been a large interest in analyzing texts, especially short texts, on different platforms. Analyzing text messages can show trends in many different industries. These trends can yield valuable insight information for companies. For example, it can tell which directions to move their business towards, and where they can find new customers.

In this Thesis, we will focus on topic models, which are statistical models used to find underlying topics in large amounts of text. We will also discuss the challenges analyzing short texts and evaluating unsupervised machine learning algorithms.

This report is written at and in collaboration with Sinch AB, which is a telecommunication company developing cloud communications platform for mobile phones. Their platform is used by their clients to send text messages to individuals. Since billions of messages are being sent every month, the company wanted us to analyze the data to get an overview of what type of messages are going through their system. Doing this, makes it possible for the company to track and predict potential trends in the telecommunication industry.

Looking at the currently existing research papers, although we found many papers discussing the process and results of using topic models on short texts, we found no papers using SMS as input. Therefore, the specific use of topic models on SMS is unique to us. This is our contribution to the field of short text clustering.

1.2 Thesis objective

The objective of this Thesis is to find the underlying topics of Sinch's dataset and the prevalence of each topic that is found. Three topic models will be evaluated based on different metrics and human evaluation in order to find the most suitable model.

1.3 Outline

In this report, Chapter 2 will first present the theoretical background used and needed to understand the report. The tools used will also be explained, together with what theory lies behind those. Afterwards in Chapter 3, we will present the methodology we used when training and finding our final models, as well as preprocessing our data. The human evaluation, which was done in the form as a survey, is also explained in the form of example questions. Then, the results obtained from our models are presented in Chapter 4, and further discussed in Chapter 5. Lastly, we present our conclusions of this thesis.

Chapter 2
Theory

2.1 Topic models

In the field of machine learning, topic modeling is a category of statistical models often used in text mining (Boyd-Graber et al., 2017). They have the property of clustering words from the same topic by making use of the word co-occurrences found in a corpus. The input to such models is generally text documents, which can range from chapters in a book to very short tweets. The model will then extract topics, represented by a list of words and their probability of belonging to a topic. The top words of such a list will form a topic interpretable by a human. An example topic would be the top words: *wind, snow, rain, sunny, cloudy, ...*, which we humans can interpret as "weather".

Using topic modeling, one can find structures in corpora. Some structures might be obvious, such as separating a newspaper into sports sections and politics. Other uses can be to find relations between research papers of different areas of subject. A concrete example would be taking each news article from a newspaper as a document, with the whole newspaper being the corpus. Topic models can then automatically separate the corpus into different topics, i.e. clusters, such as "sports" or "politics". How the topics are generated depends on the model used, and in this report, we will use latent Dirichlet allocation and biterm topic model.

2.2 Notation and terminology

In this section, we explain the notations and terminology that is used throughout the report when explaining the topic models.

The main focus of this thesis is on discrete data (text) and its basic unit *word*. We will have V unique words and use one-hot encoding to represent them. This means that a word will be represented as a unit vector that has one single element equal to one and all

other elements equal to zero. When N words are combined into a sequence, we call it a *document*. A document will be denoted as $d = (w_1, w_2, \ldots, w_N)$, where w_N is the nth word in the vocabulary. Here, we assume a bag-of-words representation, which means that the order of the word sequence does not matter. When we have a collection of M documents we call it a *corpus*. A corpus is represented as $D = \{d_1, d_2, \ldots, d_M\}$. Also, a semantic topic is what a human would refer to as a topic. Most variables are explained in Table 2.1. Note that this table is mainly a glossary for reference when reading the report later, in order to easily find the meanings of different variables used. The explanations for some of the variables will come later.

Table 2.1: List of variables and their explanations.

w_i, w_j	word
d	document
D	corpus
$\|D\|$	number of documents
$\tilde{l}$	average document length
V	number of words in the vocabulary
M	number of documents in the corpus
k	topic number
K	number of topics
ϕ	topic-word distribution
ϕ_k	topic-word distribution for topic k
$\phi_{i,j}$	probability for word w_i in topic j
θ	global document-topic distribution
θ_d	document-topic distribution for document d
$\mathbf{z}$	topic assignment for all words
z_i	topic assignment for word w_i
α	hyperparameter for document-topic distribution
β	hyperparameter for topic-word distribution
b	biterm
B	biterm multiset
$\|B\|$	number of biterms in biterm multiset

2.3 Probabilistic Building Blocks

When working with probabilistic models, the goal is to find parameters for unobserved models that can find a good way of explaining observed data. A common first step in inference is to reverse this process. By inference, we mean that we estimate the values of hidden parameters in the model. Instead of finding parameters that can make the model explain the observed data, we can instead start with generating data based on the given model parameters. Let us call this a generative story.

A generative story can be described as a recipe, listing a sequence of random events, that creates the dataset we are trying to explain. To facilitate the understanding of the models we will explain the most important probability distributions involved, how these

distributions are parameterized and what samples look like when drawn from these distributions, see Table 2.2.

Distribution	PDF	Example parameters	Example draws
Gaussian	$\frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$\mu = 2, \sigma^2 = 1.1$	$x = 2.21$
Multinomial	$\frac{\Gamma(\sum_i x_i+1)}{\prod_i \Gamma(x_i+1)} \prod_i \phi_i^{x_i}$	$\phi = [0.1\ 0.6\ 0.3]^T$	$x = 2$
Dirichlet	$\frac{\prod_{i=1}^K \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^K \alpha_i)} \prod_{i=1}^K \theta_i^{\alpha_i-1}$	$\boldsymbol{\alpha} = [1.1\ 0.1\ 0.1]^T$	$\theta = [0.8\ 0.15\ 0.05]^T$

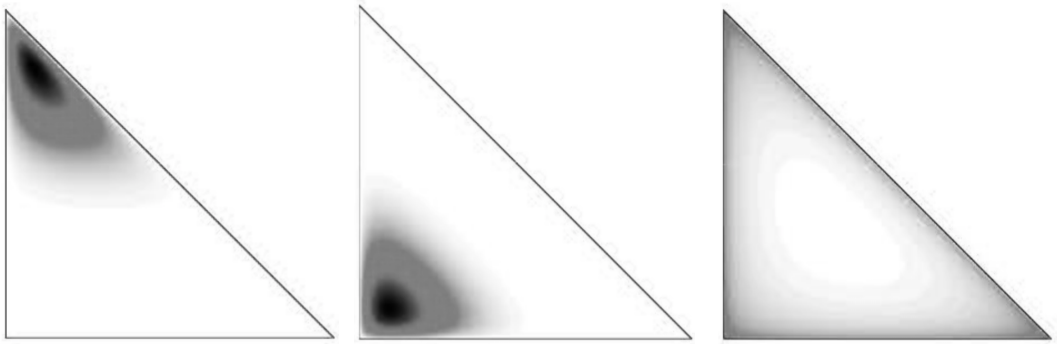
Table 2.2: Probability distributions used in the generative stories of topic models (Boyd-Graber et al., 2017). PDF stands for probability density function.

Gaussian. The Gaussian distribution does not have a central role in the topic models we are using, but we assume the reader is familiar with this distribution and will therefore include it to compare with other distributions that we are using to make it easier to understand the other distributions.

A Gaussian distribution is a continuous probability distribution. It means we can put our hand into a bag of balls and get a ball with any real value from the interval $(-\infty, \infty)$. Note, all values do not have the same probability of being picked. Gaussian distributions have two parameters, the mean μ and the variance σ^2 that control the distribution of real values in the bag. Most of the balls from the bag will have values close to the mean, and how close depends on the variance. The higher the variance is, the more spread out the samples will be. In Table 2.2, we can see example parameters for a Gaussian distribution ($\mu = 2$ and $\sigma^2 = 1.1$) and an example draw ($x = 2.21$) from this distribution.

Multinomial. As mentioned in Section 2.2, documents are a sequence of words, therefore a distribution over discrete sets is of interest. We can for example visualize the multinomial distribution by thinking of a weighted die, where each side of the die is related to a distinct probability outcome. These probabilities correspond to the parameters of a multinomial distribution. For example, in Table 2.2 we have the example parameter $\phi = [0.1\ 0.6\ 0.3]^T$. Imagine we have a die with three sides (1, 2, 3) and that each side is weighted according to ϕ. When rolling the die we can get the side with number two face up. This corresponds to the example draw $x = 2$ and the probability on position two with probability 0.6 in ϕ.

Multinomial distributions play a huge role in topic models, because they describe the relation between both words and topics, and topics and documents. The relation between words and topics is called a topic distribution, denoted as ϕ_k, where k is the topic number. Each topic gives higher weights to some words more than others. For example, we have a topic "sports" with the multinomial distribution $\phi_k = [0.1\ 0.6\ 0.3]^T$ for the words *tour*, *win* and *world*. Then, in the generative process we will explain later, we want to draw a word following the weights of the probability vector ϕ_k.



(a) $\boldsymbol{\tau} = (0.8, 0.2, 0.2)$, $\alpha_0 = 10$ **(b)** $\boldsymbol{\tau} = (0.2, 0.8, 0.2)$, $\alpha_0 = 10$ **(c)** $\boldsymbol{\tau} = (0.33, 0.33, 0.33)$, $\alpha_0 = 0.1$

Figure 2.1: Probability density functions on the $(K-1)$ simplex for different values of the base measure τ and the concentration parameter α. Darker areas correspond to higher probabilities (Boyd-Graber et al., 2017).

The relation between topics and documents is denoted as θ_d. The vector contains the probabilities of document d belonging to different topics.

For example, we have a multinomial distribution $\theta_d = [0.1\ 0.6\ 0.3]^T$ for the topics *sports*, *politics* and *economics*. Then, when drawing from θ_d in the generative process, the document created is most likely related to politics.

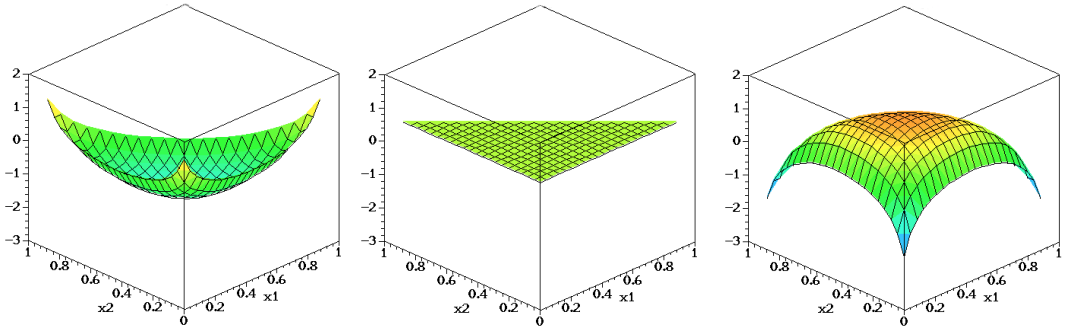
Dirichlet. In Table 2.2, we can see that the sample drawn from a Dirichlet distribution is a probability vector. More formally, a k-dimensional Dirichlet random variable θ can take values in the $(k-1)$-simplex (a k-vector θ lies in the $(k-1)$-simplex if $\theta_i \geq 0$, $\sum_{i=1}^{k} \theta_i = 1$), and has the following probability density on this simplex (Blei et al., 2003):

$$p(\theta|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} ... \theta_k^{\alpha_k - 1}, \tag{2.1}$$

where the parameter $\boldsymbol{\alpha}$ is a k-vector with components $\alpha_i > 0$, and where $\Gamma(x)$ is the Gamma function. We will see in the next section that this probability vector corresponds to the parameters for the multinomial distribution.

Similar to the Gaussian distribution, the Dirichlet distribution also has parameters that correspond to mean and variance. The expected value (mean) from a Dirichlet distribution is called the "base measure" and is denoted $\boldsymbol{\tau}$. It is the values we would get by averaging many draws from the distribution. Then we have the parameter that controls how much each draw deviates from the base measure α_0, which is called the concentration parameter. It is also common to combine $\boldsymbol{\tau}$ and α_0 into $\boldsymbol{\alpha} = \alpha_0 \boldsymbol{\tau}$ for each dimension (Boyd-Graber et al., 2017).

In Figures 2.1-2.2, we can observe the behavior of the Dirichlet distribution for different values on the concentration parameter. If α_0 is large then the draws from the distribution are close to $\boldsymbol{\tau}$. If α_0 is small, then the draws from the distribution will deviate from $\boldsymbol{\tau}$. Basically, for small values on α_0, the distribution will become very sparse, which means that only a few values have a high probability and all other values are small.



(a) $\alpha = 0.33$ **(b)** $\alpha = 1.00$ **(c)** $\alpha = 2.00$

Figure 2.2: Probability density functions in 3D with varying values on α with uniform base measure $\boldsymbol{\tau}$.

In topic modeling, the concentration parameter α_0 has a major effect on the results, because we want to create a model that can reflect the properties of the real documents. Do we want to assume a document covers all possible topics or is it more reasonable to assume only a handful topics are covered in a document? Modelling the sparsity of Dirichlet distributions is therefore of high importance.

There are other important special cases of the Dirichlet distribution that is good to mention. When $\boldsymbol{\tau}$ is uniform, we say we have a *symmetric* distribution. This means that all topics are equally favored.

In Table 2.2, we can see an example parameter ($\boldsymbol{\alpha} = [1.1\ 0.1\ 0.1]^T$) of a Dirichlet distribution. The weights in the example are not uniform, which means some topics or words have been given more weight. When drawing a sample from the distribution with α as parameter, we get a new vector that corresponds to our word or topic distributions ($\theta = [0.8\ 0.15\ 0.05]^T$) .

In the rest of the report, we do not have prior knowledge of any words or topics of the dataset we are working with, and therefore we use a symmetric Dirichlet distribution.

2.4 Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is a generative probabilistic model created by Blei et al. (2003). Such generative probabilistic models can be described by a generative process. The generative process for the LDA model we use is given below:

1. For each topic k:
   (a) draw a topic-word distribution $\phi_k \sim$ Dirichlet(β)
2. For each document d in D:
   (a) draw a document-topic distribution $\theta_d \sim$ Dirichlet(α)
   (b) For each word w_i in d:
      i. draw a topic assignment $z_i \sim$ Multinomial(θ_d)
      ii. draw a word $w_i \sim$ Multinomial(ϕ_{z_i})

Table 2.3: Example of LDA output

"airport security"	"competitive sports"	"court"
security	win	man
air	lead	charge
service	take	court
airport	world	face
sydney	title	murder
flight	aussie	jail
concern	tour	trial
new	open	accuse
plan	record	sentence
fear	claim	drug
...	...	...

In the generative process, we use "~" to describe drawing a sample from a distribution. For instance, "$\phi_k \sim$ Dirichlet(β)" means that we draw a sample $\phi_k = [0.1, 0.05, 0.001, ..]$ from a (symmetric) Dirichlet distribution with the parameter β.

The intuitive interpretation for α can be found in the Dirichlet paragraph above. Meanwhile, β controls the word density of each topic. A high value of β means that the probability of each word being chosen is split more evenly among each word. This effectively results in having more dominant words in the topic. On the other hand, a low β will result in allowing fewer dominant words. The goal is to find a β which yields one semantic topic to appear in each topic cluster.

In Eq. 2.2 we have the resulting joint distribution from the generative process described above.

$$p(\theta, \phi, \mathbf{z}, d|\alpha, \beta) = p(\phi|\beta)p(\theta|\alpha)p(\mathbf{z}|\theta)p(d|\phi_z) \tag{2.2}$$

The variables that are most interesting to us are the latent (i.e. hidden) variables, $\mathbf{z}$, θ and ϕ, where $\mathbf{z}$ is the topic assignment for all words.

What makes LDA very interesting for our problem is that it can in an unsupervised way learn topics that correspond to the human interpretation. In Table 2.3, we see an example of three topics. Each column represents one topic. The top row is a summary of each topic, added afterwards by humans.

Plate model. Plate models are a graphical way to represent the relations between random variables. In our case, we have a plate model of LDA in Figure 2.3, which shows the relations between LDAs different random variables. Each circle represents a random variable in the model. A circle with gray background means that it is observable. Meanwhile, a white background means that it is a latent variable. As for the plates, they represent a certain number of repetitions, given by a fixed variable. Lastly, the arrows show the dependencies between the random variables. For example, the latent variable θ is drawn M times (once for each document), based on the random variable α. We also see that the topic assignment z is dependent on the latent variable θ. Even though α is a random variable, we choose to set it ourselves as a hyperparameter.

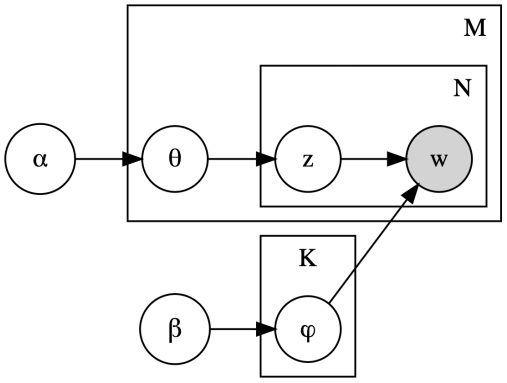


Figure 2.3: Plate model representation of LDA.

It is worth checking the generative process for LDA again, while comparing it to Figure 2.3. One can then see the similarities, and that it is a helpful tool to visualize the generative process of LDA.

2.4.1 Inference

In topic modelling, the key problem is to reverse the generative process and compute the posterior distributions of the latent variables in the model given the observed data (Darling, 2011). Focusing on LDA, this means solving the following equation:

$$p(\theta, \phi, \mathbf{z}|d, \alpha, \beta) = \frac{p(\theta, \phi, \mathbf{z}, d|\alpha, \beta)}{p(d|\alpha, \beta)} \tag{2.3}$$

Since the normalization factor, $p(d|\alpha, \beta)$, cannot be computed exactly, the distribution is in general *intractable* to compute (Blei et al., 2003). However, even though exact inference is intractable for the posterior distribution, there are a number of approximate inference algorithm that can be considered. In this report, we use a Markov chain Monte Carlo (MCMC) algorithm called Gibbs sampling.

2.4.2 Gibbs sampling

The goal of MCMC algorithms is to construct a Markov chain that has the posterior distribution in Eq. 2.3 as its stationary distribution. Sampling from the converged distribution should be close to sampling from the desired posterior distribution (Darling, 2011). In general the following procedure is being performed when using Gibbs sampling:

1. Randomly initialize each topic assignment z_i
2. For each iteration t:

   2.1 $z_1^{t+1} \sim p(z_1|z_2^{(t)}, z_3^{(t)}, ..., z_m^{(t)})$

   2.2 $z_2^{t+1} \sim p(z_2|z_1^{(t+1)}, z_3^{(t)}, ..., z_m^{(t)})$

   2.m $z_m^{t+1} \sim p(z_m|z_1^{(t+1)}, z_2^{(t+1)}, ..., z_{m-1}^{(t+1)})$

Instead of probabilistically sampling a new state all at once for all variables, the idea behind Gibbs sampling is to do it one at a time. When probabilistically sampling a new state for a variable, the new state will depend on all other variables states (Resnik and Hardisty, 2009). This procedure is repeated several times until convergence has been reached. In theory, convergence is guaranteed with Gibbs sampling but in practice it is not possible to know how many iterations are required to reach the stationary distribution. Gibbs sampling is a very powerful tool with good performance. Even though not knowing when it converges is a big problem, one can obtain an acceptable estimation of convergence by calculating the log-likelihood (Darling, 2011). We will discuss more about log-likelihood further on in the report.

An example of an iteration with Gibbs sampling with three variables is represented below (Resnik and Hardisty, 2009):

- The new value of z_1 is sampled conditioning on the old values of z_2 and z_3.
- The new value of z_2 is sampled conditioning on the *new* value of z_1 and the *old* value of z_3.
- The new value of z_3 is sampled conditioning on the *new* values of z_1 and z_2.

2.4.3 Collapsed Gibbs sampling.

For LDA, we are interested in the topic assignments, document-topic distributions and topics. Before explaining collapsed Gibbs sampling, we will talk about these variables more thoroughly.

Topic Assignments. As mentioned in Section 2.4, every word is assumed to be generated from a topic, therefore we can consider the topic assignment of a word, z_i, to be a variable. For example, if we take a look at the word "court", this word can refer to a tennis court or a court where a trial can be held. In other words, a word can have different meanings and therefore appear in different topics, i.e. have different topic assignments. This variable is important to be able to estimate the global properties of the topic model (Boyd-Graber et al., 2017).

Document-topic distributions. Document-topic distribution is a distribution over topics for a document. Basically, it tells us how popular each topic is in a document. Counting how often a topic appears in a document can give us a hint of its popularity. Let's define $n_{d,k}$ as the number of times topic k appears in document d. The more popular the topic is in the document the larger $n_{d,k}$ will be. Since we are working with probabilities, and the variable is larger than one, we can normalize it with the number of words in the document (Boyd-Graber et al., 2017).

$$\frac{n_{d,k}}{\sum_{k'} n_{d,k'}} \tag{2.4}$$

Observe that Eq. 2.4 can sometimes be problematic, because if a topic does not occur in a document the numerator will be equal to zero. The equation is ignoring the influence of the Dirichlet distribution, therefore the following ratio is preferred

$$\theta_{d,k} = \frac{n_{d,k} + \alpha_k}{\sum_{k'} n_{d,k'} + \alpha_{k'}} \tag{2.5}$$

This ratio will never become zero (since $\alpha > 0$), and that is good because we want all topics to have the probability to appear in a document. This opens up the door for the sampler to explore more of the possible combinations. In Eq. 2.5, we can see how α affects the document-topic distribution. α effectively adds additional counts to each topic in each document. This means that the actual counts to each topic matter less for higher α. Instead, the result will approach a uniform distribution as α increases. On the other hand, a low α will let the documents contribute on their own.

Topics. A topic is a distribution over words, also referred to as topic-word distribution in the report. In order to understand what a topic is about we have to look at the words that have been assigned to that topic. The probability of a word in a topic is defined as

$$\phi_{k,w} = \frac{n_{k,w} + \beta_w}{\sum_{w'} n_{k,w'} + \beta_{w'}} \tag{2.6}$$

where $n_{k,w}$ is the number of words w in topic k and β is the Dirichlet parameter for the topic-word distribution (Boyd-Graber et al., 2017). Similarly to Eq. 2.5 and α, Eq. 2.6 reveals the effect of β on the topic-word distribution. A high value of β adds more additional counts to each word in each topic. In that case, the actual contribution from the document matters less. A low value of β, however, will just like for α and the document-topic distribution, let the documents contribute the most. Still, due to $\beta > 0$, it will always guarantee that all words have a small probability of appearing in any topic.

As we can see Eqs. 2.5-2.6, θ_d and ϕ_k depend on z_i. This is because we need to know the topic assignments $\mathbf{z}$ in order to count $n_{d,k}$ and $n_{k,w}$. Knowing this makes it possible to use a simpler algorithm by integrating out the multinomial parameters and only sample z_i. This is called a *collapsed* Gibbs sampler. Note that the topic assignment z is a latent variable, drawn from a multinomial distribution according to the generative process, while the topic k is a concrete value of z, i.e. $k \in 1, 2, .., K$ (Darling, 2011).

The collapsed Gibbs sampler calculates the probability of a topic z being assigned to a word w_i, given the topic assignments of all other words. More formally, the algorithm will compute the following posterior probability

$$p(z_i = k|\mathbf{z_{-i}}, d, \alpha, \beta) = \frac{p(z_i, \mathbf{z_{-i}}, d|\alpha, \beta)}{p(\mathbf{z_{-i}}, d|\alpha, \beta)} \tag{2.7}$$

where $\mathbf{z}_{-i}$ means all topic assignments except for z_i and d all words in a document. The specific Gibbs sampling equation for assigning a word to a particular topic for LDA can be found in Eq. 2.8. If the reader is interested in the derivation steps they can be found in Darling (2011).

$$p(z_i = k|\mathbf{z_{-i}}, d, \alpha, \beta) \propto \frac{n_{d,k}^{(-i)} + \alpha_k}{\sum_{k'} n_{d,k'}^{(-i)} + \alpha_{k'}} \frac{n_{k,w}^{(-i)} + \beta_w}{\sum_{w'} n_{k,w'}^{(-i)} + \beta_{w'}} \tag{2.8}$$

$n_{d,k}^{(-i)}$: Number of times topic k occurs in document d
$n_{k,w}^{(-i)}$: Number of times word w occurs in topic k
α_k : Dirichlet parameter for document-topic distribution
β_w : Dirichlet parameter for topic-word distribution

For every word, we compute Eq. 2.8 for all topics. This results in a word-topic distribution. From this multinomial distribution, we draw a new topic assignment for word w_i, update $n_{d,.}$ and $n_{.,w}$ and move on to the next word and repeat. Further on, collapsed Gibbs sampling is referred to as Gibbs sampling.

Log-likelihood. Log-likelihood is a standard measure of performance for statistical models of natural language and is defined as (Pleplé, 2013):

$$L(D_{test}) = \log p(D_{test}|\boldsymbol{\phi}, \alpha) = \sum_{d} \log p(d_u|\phi, \alpha), \tag{2.9}$$

where D_{test} is a collection of unseen documents d_u and ϕ is the word distribution matrix. The higher the log-likelihood value, the better the model.

2.4.4 Gibbs sampling example

Let's illustrate with an example what exactly is happening when performing the Gibbs sampling algorithm.

Assume we observe a document of length 5 created from the generative process, with random topic assignments *3,2,3,1,1*. From each topic, we sample the words *home, lending, boosts, bank, profits*, see Table 2.4. Assume the same procedure is repeated multiple times for all documents in the dataset.

Table 2.4: Example document with topic assignments and sampled words from the generative process.

3	2	3	1	1
home	lending	boosts	bank	profits

Next, we want to improve the topic assignments of the words in the documents. For example, in Table 2.5, *home* has occurred 36 times and was assigned topic 3, 35 times and topic 1, once.

Table 2.5: Topic assignments of words in example corpus.

	1	2	3
bank	42	1	0
boosts	0	0	20
home	1	0	35
lending	10	8	1
profits	50	0	1
...			

When we want to update the topic assignment of a word we start with pretending that the topic assignment of that word is unknown. Let's start with the word *lending* as in Table 2.6.

Table 2.6: Assume the topic assignment of *lending* is unknown.

3	?	3	1	1
home	lending	boosts	bank	profits

Due to assuming not knowing the topic assignment of this word, the number of times the word *lending* occurs in topic 2 decreases from 8 to 7, see Table 2.7.

Table 2.7: The number of times *lending* occurs in topic 2 decreases from 8 to 7.

	1	2	3
bank	42	1	0
boosts	0	0	20
home	1	0	35
lending	10	**7**	1
profits	50	0	1
...			

In the next step, we want to use the conditional probability distribution from Eg. 2.8. Remember, the first factor tells us how much a document likes a topic. We can visualize it with horizontal bars, see Figure 2.4.

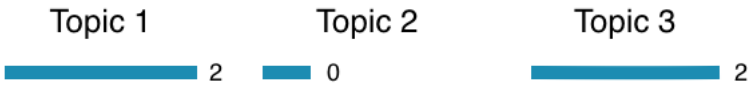


Figure 2.4: Topic distribution for the document represented as horizontal bars.

As we can see, the document uses topic 1 and 3 more compared to topic 2. Note that topic 2 is not completely zero due to the Dirichlet parameter in Eq. 2.8. The second factor of the equation tells us how much a topic likes a word. Once again this can be visualized with vertical bars, see Figure 2.5.

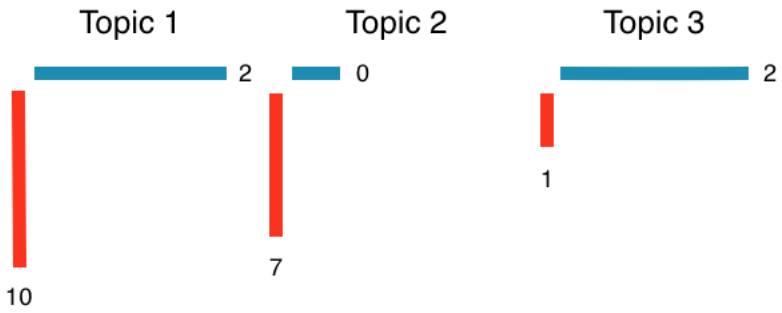


Figure 2.5: The number of times *lending* occurs in each topic represented as vertical bars.

We can see that the word *lending* is used a lot in topic 1, a fair bit in topic 2 and barely in topic 3. Knowing the horizontal bars corresponds to the first factor and the vertical bars to the second factor allows us to think about this geometrically.

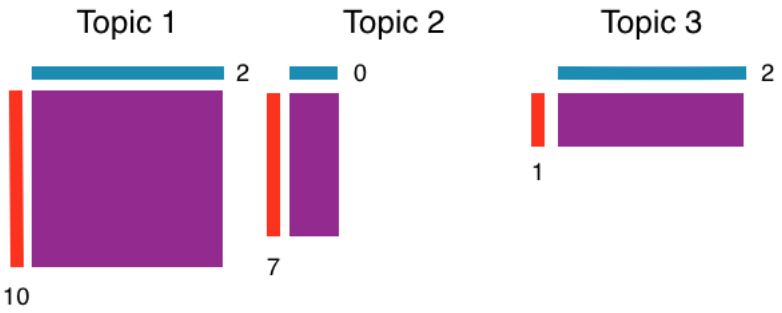


Figure 2.6: Geometrical interpretation of topic assignment probabilities.

Multiplying the bars together as in Figure 2.6 creates the rectangular areas, which corresponds to the values of the conditional probabilities. In this case, the conditional probability for selecting topic 1 is very large.

Let's say topic 1 is being selected as the topic assignment for the word *lending* as in Table 2.8, then the tables will be updated accordingly to Table 2.9.

Table 2.8: Topic assignment 1 has been assigned to *lending*.

3	1	3	1	1
home	lending	boosts	bank	profits

For each iteration in the Gibbs sampling algorithm, these steps are performed for *all* words in *all* documents.

2.4.5 Biterm Topic Model

Biterm topic model (BTM) is a topic model based on forming biterms from a corpus, proposed by Yan et al. (2013). According to their own tests, it performed better than LDA

Table 2.9: The number of times *lending* occurs in topic 1 increases from 10 to 11.

	1	2	3
bank	42	1	0
boosts	0	0	20
home	1	0	35
lending	**11**	7	1
profits	50	0	1
...			

on short texts. The model assumes a generative process from which a biterm multiset is formed. This is an alternative representation of the corpus we observe, and we wish to infer the latent variables of this process from the biterm multiset.

We can write the generative process for BTM as:

1. For each topic k
   (a) draw a topic-word distribution $\phi_k \sim$ Dirichlet(β)
2. Draw a topic distribution $\theta \sim$ Dirichlet(α) for the whole collection
3. For each biterm b in the biterm multiset B
   (a) draw a topic assignment $z_b \sim$ Multinomial(θ)
   (b) draw two words: $w_i, w_j \sim$ Multinomial(ϕ_{z_b})

Here, we can see a difference compared to LDA, which draws a document-topic distribution for every document. By forming biterm from all documents and storing them in the biterm multiset B, we alleviate the problem of sparsity in data, which LDA is known to have problems with in short texts (Yan et al., 2013). Going back to BTM and its generative process, the probability of each biterm is given by Eq. 2.10:

$$P(b) = \sum_z P(z)P(w_i|z)P(w_j|z) = \sum_z \theta_z \phi_{w_i|z} \phi_{w_j|z} \tag{2.10}$$

where $P(z)$ is the probability of a topic being chosen and $P(w_i|z)$ or $P(w_j|z)$ is the probability of a word being chosen, given topic z. Also, by assuming independence between biterms, a corpus, which is equivalent to the biterm multiset, can be represented by Eq. 2.11:

$$P(B) = \prod_{i,j} \sum_z \theta_z \phi_{w_i|z} \phi_{w_j|z}, \tag{2.11}$$

In reality, since these probability distributions in Eq. 2.11 are unknown, they are estimated from counting the words in a biterm multiset formed from the corpus.

Before training, BTM extracts biterms from all documents. This is done by taking an unordered Cartesian product. In other words, this basically means that every unordered pair of words in a document is paired together into a biterm. For example, the sentence *I ate the fruit.* would generate the following biterms:

I ate, I the, I fruit, ate the, ate fruit, the fruit

By forming biterms from each document into a biterm multiset covering the whole corpus, the problem of sparsity during topic inference which exists for LDA in short texts will be made up for. Because the purpose of biterms is to capture the local word co-occurrences, a window size is introduced during the biterm generation. In this way, words that occur too far apart from each other will not form a biterm.

As BTM models the biterms over the whole corpus directly, there is no direct modeling of the document-topic distribution as in LDA. Instead, only the topic-word distribution and the global topic distribution are given. However, the document-topic distribution can be approximated by marginalizing over biterms, as shown in Eq. 2.12:

$$P(z|d) = \sum_b P(z|b)P(b|d) \tag{2.12}$$

The first term in the sum can then be considered as the probability of choosing topic z times the probability of independently choosing the two words of the biterm b in that topic, then normalized. This can be seen in Eq. 2.13. As for $P(b|d)$, this can be chosen empirically, and is very close to a uniform distribution, an estimation which works well in practice.

$$P(z|b) = \frac{P(z)P(w_i|z)P(w_j|z)}{\sum_{z'} P(z')P(w_i|z')P(w_j|z')} \tag{2.13}$$

Hyperparameters. BTM has four hyperparameters which play a big role in determining the final model. The number of topics K is one which has to be decided upon, as well as α, β and the number of Gibbs iterations until the model is trained. Similarly to LDA, α and β control the distribution of topics in documents and the distribution of words in topics respectively. Since α and β are hyperparameters of symmetric Dirichlet distributions, a higher α means that the topic assignments for each biterm will be more evenly distributed. Similarly for β, a higher value means that more words are likely in each topic. Typically in short texts, we expect very few topics in each document, and the keywords that describe each topic should not be too many. Therefore, fairly low values of α and β are to be expected.

Plate model. The generative process assumed by BTM can be represented in a plate model, just as LDA, seen in Figure 2.7. In the plate model, we can see that β is the hyperparameter for φ, and that φ is drawn K times. This is the topic-word distribution for each topic. We can also see that θ is only drawn once, using α as a hyperparameter. From θ, we draw $|B|$ biterms, beginning with the topic assignment z, then moving on to the two words w_i and w_j that form the biterm. This plate diagram can be compared with the generative process of BTM, where similarities can be found. Once again, like in LDA, collapsed Gibbs sampling can be used to sample the topic assignment and infer the latent variables ϕ and θ separately afterwards.

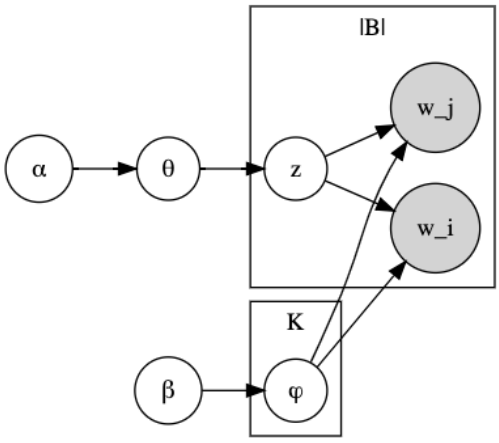


Figure 2.7: Plate model representation of BTM.

Gibbs sampling. In Eq. 2.14, we see the resulting joint distribution from BTM's generative process.

$$p(\theta, \phi, \mathbf{z}, b|\alpha, \beta) = p(\phi|\beta)p(\theta|\alpha)p(\mathbf{z}|\theta)p(b|\phi_z) \tag{2.14}$$

Just like in LDA, the interesting variables are $\mathbf{z}, \theta$ and ϕ. However, what we actually want to solve is Eq. 2.15, since we observe the dataset.

$$p(\theta, \phi, \mathbf{z}|b, \alpha, \beta) = \frac{p(\theta, \phi, \mathbf{z}, b|\alpha, \beta)}{p(b|\alpha, \beta)} \tag{2.15}$$

However, the denominator is intractable once again, so we turn to Gibbs sampling. The algorithm for Gibbs sampling in BTM consists of initializing all biterms with a topic assignment, then for each biterm in the biterm multiset, removing the topic assignment of a biterm and recompute it based on all other topic assignments. This conditional probability can be seen in Eq. 2.16:

$$P(z_b|\mathbf{z}_{-b}, B, \alpha, \beta) \propto \frac{(n_k^{(-i)} + \alpha)}{\sum_{k'} n_{k'}^{(-i)} + \alpha} \cdot \frac{(n_{k,w_i}^{(-i)} + \beta)(n_{k,w_j}^{(-i)} + \beta)}{(\sum_{w'} n_{k,w'}^{(-i)} + \beta)^2} \tag{2.16}$$

$n_k^{(-i)}$ is the number of times the biterm b appears in the corpus excluding the current biterm itself, while $n_{k,w_i}^{(-i)}$ and $n_{k,w_j}^{(-i)}$ is the number of times word w_i and w_j respectively appears in topic k, once again excluding the words in the biterm b.

Recomputing the topic assignments for all biterms is one iteration of Gibbs sampling. This process is then repeated, and will converge towards the actual distribution of the latent variables of the generative process.

Once the Gibbs sampling is done, figuring out the latent variables is just a matter of counting the topic assignments of the biterms according to Eqs. 2.17 and 2.18:

$$\theta_k = \frac{n_k + \alpha}{|B| + K\alpha} \tag{2.17}$$

$$\phi_{k,w} = \frac{n_{k,w} + \beta}{\sum_{w'} n_{k,w'} + \beta} \tag{2.18}$$

Table 2.10: Time and memory complexity of the topic models used.

Method	Time complexity	Memory complexity
LDA	$O(K\|D\|\tilde{l})$	$\|D\|K + VK + \|D\|\tilde{l}$
BTM	$O(K\|B\|)$	$K + VK + \|B\|$

Once again, the variables here are the same as in Eq. 2.16, though no word counts are excluded. In Eqs. 2.16-2.18, we see the presence of the hyperparameters α and β. They can be viewed as additional counts of each word and topic. Here, we see that assigning a high value of these hyperparameters will yield topics with more words being dominant and increase the prevalence of each topic in the documents. Also, since the hyperparameters are strictly positive, there will always be an additional count added to each word. Having the additional counts allows each word to have some probability of appearing in each topic, even if it is minimal. As a consequence, it allows for the model to explore more possible combinations of topics for each biterm and in turn, also documents.

2.4.6 Time and memory complexity

According to Yan et al. (2013), the complexity of the two methods are given in Table 2.10.

K is the number of topics, $|D|$ is the number of documents, $\tilde{l}$ is the average document length, V is the number of unique words in the corpus and $|B|$ is the size of the biterm multiset from BTM. The time consuming part of the algorithms are the conditional probability of Gibbs sampling seen in Eq. 2.16 for BTM and Eq. 2.8 for LDA. The rest of the algorithm is mostly initialization as well as some final estimations of the latent variables, i.e. θ and ϕ, which only needs to be done once. Memory-wise, BTM needs a counter for n_k, the number of times a topic appears in total, resulting in K counters. It also needs to keep track of $n_{k,w}$, the number of times each word appears in each topic, which means additional VK counters. Finally, each biterm has a topic assignment which needs to be tracked as well. This requires $|B|$ counters.

LDA requires counters for $n_{d,k}$, the number of times each topic appears in each document. That will be $|D|K$ counters. The number of times each word appears in each topic $n_{k,w}$ also requires VK counters just like in BTM. Lastly, the topic assignment z for each word occurrence requires $|D|\tilde{l}$ counters.

2.5 Visualization

The main visualization tool we use is called pyLDAvis, a Python port of LDAvis for R (Sievert and Shirley, 2014). Despite its name, it allows for visualization of topic models in general, given the right input. This includes the topic-word distribution and the document-topic distribution of the topic model. The visualization tool then makes use of a dimensionality reduction technique to bring down the topics into a 2D representation. It also manages to preserve some "closeness" in semantic meaning of topics during this process, meaning that topics in the same field can be found close to each other. This means

that topics such as court and crime lie closer to each other compared to golf, which lies closer to for example cricket.

The options for dimensionality reduction of pyLDAvis is limited to three algorithms, namely principal coordinate analysis (PCoA), metric multidimensional scaling (mMDS) and t-distributed stochastic neighbor embedding (t-SNE). In this report, we use PCoA as the dimensionality reduction technique. The method requires a distance measure, which we choose to be the Jensen-Shannon distance. This distance can be used to measure similarities between two probability distributions. It is shown in Eq. 2.19:

$$D_{JS}(P, Q) = \sqrt{\frac{D_{KL}(P \parallel M) + D_{KL}(Q \parallel M)}{2}} \tag{2.19}$$

Where M is the pointwise mean of P and Q, and $D_{KL}(\cdot, \cdot)$ is the Kullback-Leibler divergence, given in Eq. 2.20:

$$D_{KL}(P, Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)} \tag{2.20}$$

where $P(x)$ and $Q(x)$ are probability distributions. After computing all the distances and storing them in a matrix, a principal component analysis (PCA) is then performed on this distance matrix. PCA is a generalization of diagonalizing a matrix (Abdi and Williams, 2010). Picking the two largest principal components of this distance matrix, they can be plotted by projecting all cluster centers onto the plane spanned by the chosen principal components. The principal components correspond to the eigenvectors of a diagonalization. This is the result from pyLDAvis. An example of how pyLDAvis looks like can be seen in Figure 2.8 and 2.9.

Since pyLDAvis depicts topics, which consists of words, there is no direct relation between the size of each topic (circle) and the number of documents in them. Instead, the marginal topic distribution is dependent on which words belong to each topic and the probability of the word belonging to the topic.

Looking at Figure 2.8, one can see the distribution of topics in the two principal components to the left. The chosen topic is marked in red. To the right, the red bar shows how many times a word occurs in the chosen topic. Meanwhile, the blue bar represents the overall word frequency in the whole corpus.

There are two key metrics in pyLDAvis which are used, namely *relevance* and *saliency*. The definition of relevance is given in Eq. 2.21:

$$relevance(w, k|\lambda) = \lambda \log(\phi_{k,w}) + (1 - \lambda) \log \frac{\phi_{k,w}}{P_w} \tag{2.21}$$

where $\phi_{k,w}$ is the probability of word w_i in topic k and P_w is the overall probability of word w_i in the corpus. Reasoning about Eq. 2.21, one can see that the relevance takes both absolute frequencies as well as relative frequencies into account. The extreme cases of $\lambda = 0$ and $\lambda = 1$ showcases what each term does. When $\lambda = 0$, what matters is the quotient between the probability of a word occurring in the topic against the probability of the word appearing globally in the corpus. Graphically, this means that the top words of each topic are sorted in order by the quotient between the red bar and the blue bar to the right in Figure 2.8. The other extreme case when $\lambda = 1$ implies that the words are sorted by the length of the red bar to the right in Figure 2.8 for each topic. For values of λ

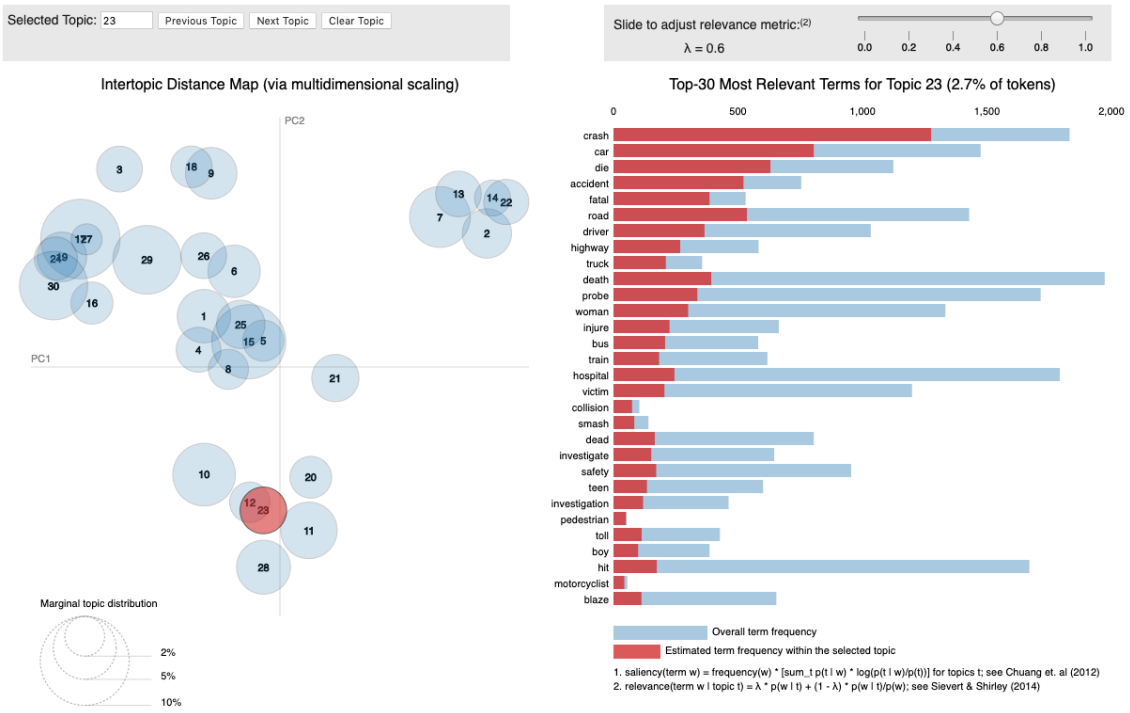


Figure 2.8: pyLDAvis showing a topic about car crashes from one million headlines (Yadav, 2018).

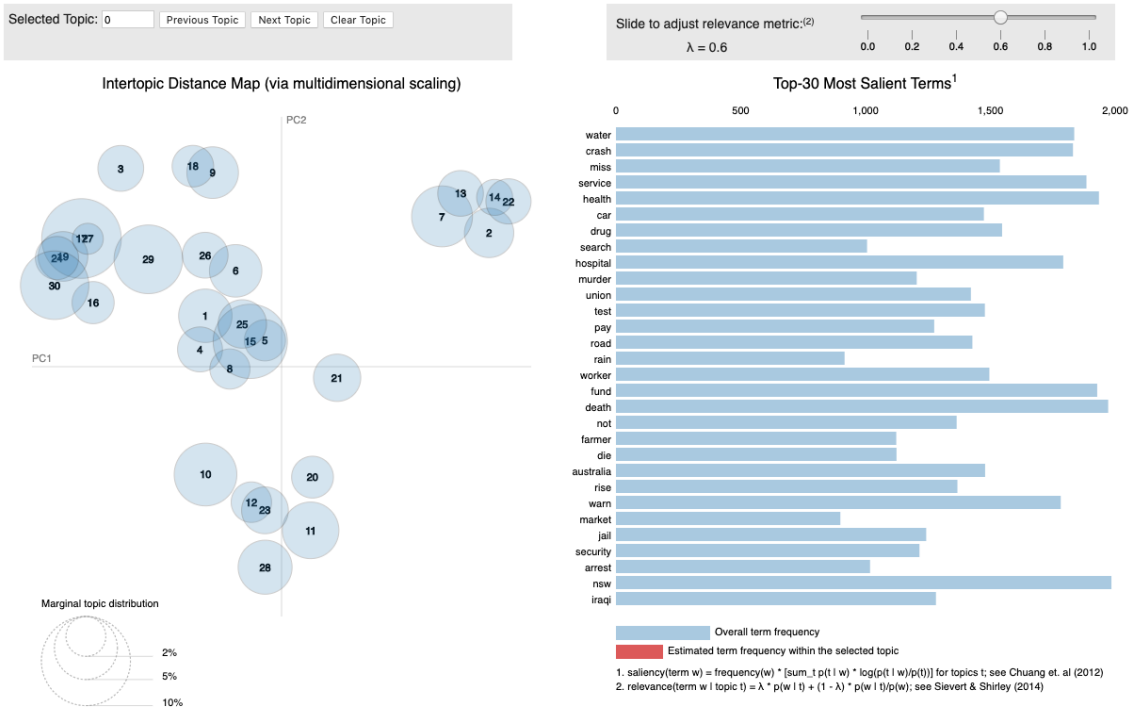


Figure 2.9: pyLDAvis showing the corpus-wide most salient terms according to Eq. 2.22.

between 0 and 1, pyLDAvis will take both absolute and relative frequencies into account,

weighing them according to λ.

$$saliency(w) = P(w) \sum_{k} P(k|w) \log \frac{P(k|w)}{P(k)} \tag{2.22}$$

where $P(k|w)$ is probability of topic k given word w_i and $P(k)$ is the probability of topic k.

We can see that the saliency is defined as the overall frequency of word w_i times the Kullback-Leibler divergence between a topic k and a word w_i. The Kullback-Leibler divergence here describes how informative a specific word is to a topic (Chuang et al., 2012). The most salient terms of a news dataset can be seen in Figure 2.9. A word which occurs in many topics will get a lower value than words only occurring in specific topics. Of course, the overall word frequency will also play a role in determining the most salient terms.

However, since topics are not directly related to documents, there is still a need for evaluation methods.

2.6 Evaluation

There are multiple ways of evaluating a topic model. Human evaluation where we design tests is one possibility, and we will use two variants of these, called word intrusion and topic intrusion (Chang et al., 2009). These will be presented further on in the Section 3.5. Another possibility is to use different kinds of metrics such as log-likelihood. It has however been shown that log-likelihood has a negative correlation with human evaluation (Chang et al., 2009). Even though there are other metrics that has positive correlation, there is no guarantee that optimizing topic models over such metrics will yield an optimal human-interpretable topic model, especially when taking the resolution of topics into account.

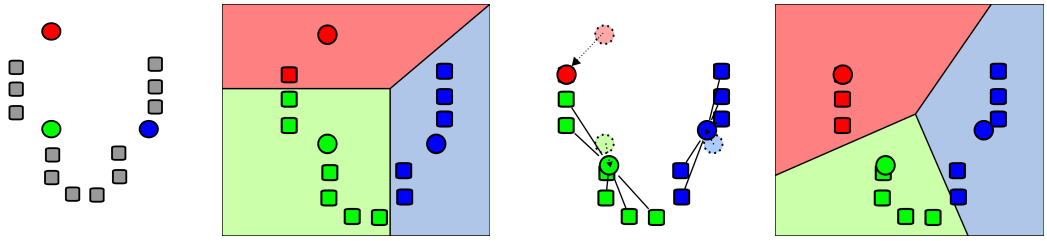
However, since there are metrics that have a positive correlation with human evaluation, they can still be used as a guideline (Röder et al., 2015). Among the metrics that are used, there is normalized pointwise mutual information (NPMI) and UMass. These are coherence scores which can be used to measure how good a topic is. Röder et al. (2015) provides a framework for evaluating topic models. Among these are some previously established measurements such as UMass and NPMI. There, they are defined as seen in Eqs. 2.23 and 2.24.

$$C_{UMass} = \frac{2}{N \cdot (N-1)} \sum_{i=2}^{N} \sum_{i-1}^{j=1} \log \frac{P(w_i, w_j) + \epsilon}{P(w_j)} \tag{2.23}$$

$$NPMI(w_i, w_j) = \frac{\log \frac{P(w_i, w_j)+\epsilon}{P(w_i) \cdot P(w_j)}}{-\log(P(w_i, w_j) + \epsilon)} \tag{2.24}$$

where the denominator of Eq. 2.24 is a normalization factor, such that the value is limited between [-1,1].

The sum in C_{UMass} is computed over the top words of each topic, a number which has to be specified. Meanwhile, the NPMI is computed for each pair of top words and summed up, then averaged. However, there is still the question of how the probabilities in Eqs. 2.23 and 2.24 are computed. One difference is that C_{UMass} uses Boolean document counts. This



(a) Placing initial cluster centers

(b) Assign cluster to data points.

(c) Move cluster centers.

(d) Repeat **(b)** and **(c)** until convergence.

Figure 2.10: Example of how K-means algorithm works. The circles are cluster centers and the squares are data points.

means that one count is added for each document which contains word w_i (and w_j). NPMI instead uses a sliding window. This means that each time w_i and w_j appears within n words of each other in a document, the joint probability $P(w_i, w_j$ increases by one. Meanwhile, the probability for a single word is still the number of times it appears in the corpus.

However, since coherence scores do not have any units, it becomes harder to compare them. For example, a difference in NPMI coherence score of -0.9 from one model compared to -0.3 from another is hard to quantify. If both models work with the same dataset, we can say that the second model is better than the first, but not really by how much. For that, we would need to manually examine each topic and evaluate the model ourselves.

Fortunately, there is also another way to evaluate our models. Since each document has a document-topic distribution, we can use these to classify the documents. The simplest way of doing this is by assigning the topic which is the most dominant for each document. By dominant, we mean the topic which has the highest probability for a document in the document's document-topic distribution. We call this method the dominant topic.

Another method of assigning labels to documents is to use K-means on each document. The representation of each document will be its document-topic distribution. The hyperparameter for K-means is the number of clusters K. The algorithm will then initialize by randomly placing K cluster centers. Then, each data point will be assigned a label corresponding to the cluster center which lies closest to it. For each label, a new cluster center is computed. This center is placed at the center of mass of all points belonging to that label (assuming unit weight for each point). The process of assigning data points a label and then computing new cluster centers is iterated for a set number of times or until convergence. An example of how K-means works can be seen in Figure 2.10:

Having classified documents using two methods, one can compare these by assuming that one of the methods is the correct answer and another method is the prediction. Then, using the F1-score one can see similarities and dissimilarities between the different methods. The F1-score is the harmonic mean of the precision and recall. All these are defined in Eqs. 2.25-2.27:

$$recall = \frac{True\ positives}{True\ positives + False\ negatives} \tag{2.25}$$

$$precision = \frac{True\ positives}{True\ positives + False\ positives} \tag{2.26}$$

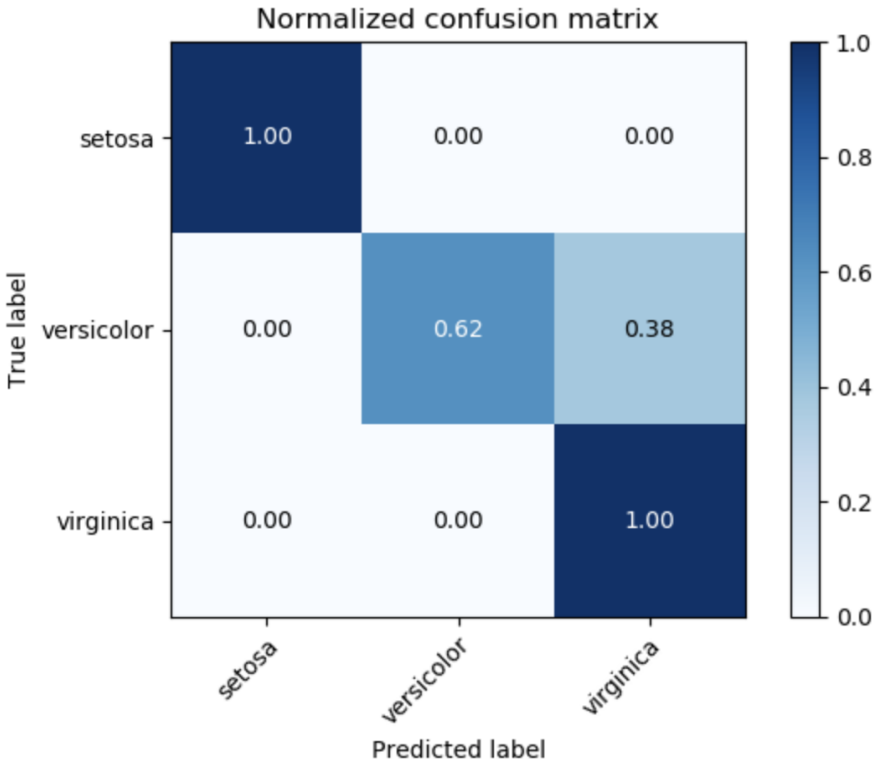


Figure 2.11: Example of a normalized confusion matrix, taken from scikit-learn (Pedregosa et al., 2011).

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.27}$$

Given a specific topic k, the true positives are the documents that are correctly classified into topic k. The false negatives are documents that were classified in another topic while actually belonging to topic k. Lastly, the false positives are documents that were classified in topic k while actually belonging to another topic.

On top of the F1-scores for each topic, a confusion matrix can be plotted to get an overview of how similar the different methods classify the documents. A confusion matrix shows the relation between the actual label and the predicted label, sorted in a matrix. This can also be normalized to get a feel for how correct the classifications are, see Figure 2.11.

2.6.1 Collocations

In some cases, there are words that do not make sense by themselves, or change meaning when they are next to each other. Word that often appear together are called collocations. When doing topic modelling, we want our models to recognize collocations. Seeing the words *world* and *cup* by themselves can give them different meanings than if they appear together as *world cup*. Similarly, seeing the words *Sri* and *Lanka* by themselves without the other one may confuse people. *Sri Lanka*, on the other hand, is meaningful.

Working with our corpus, we need a method to systematically find collocations. It is reasonable to assume that collocations are formed from word pairs that often appear

together and seldom alone. In other words, if they have a high ratio of word co-occurrence compared to appearing alone, they are likely to be a collocation. A good example of this is *Sri Lanka*, whose words one rarely sees appearing alone. Since word co-occurrence is a suitable measure for this, one can use NPMI from Eq. 2.24. By counting the number of times the words occur alone and the number of times they appear together, we can compute the NPMI score of that word pair. After doing this, one can then limit the collocations that will form by setting a threshold, only allowing word pairs with a NPMI score higher than the threshold to actually form collocations.

Chapter 3
Method

In this chapter the applied methodology for LDA, BTM and LDA-U is presented. The first section introduce the tools and the dataset we have used. Then, we explain how we take advantage of the dataset to create a new input for one of the models, and how the most significant parameters are being selected. In the following sections, the preprocessing steps are described and also shown in Figure 3.1. Then, we describe the way the topic clusters are being evaluated. Lastly, the way the messages were classified is explained.

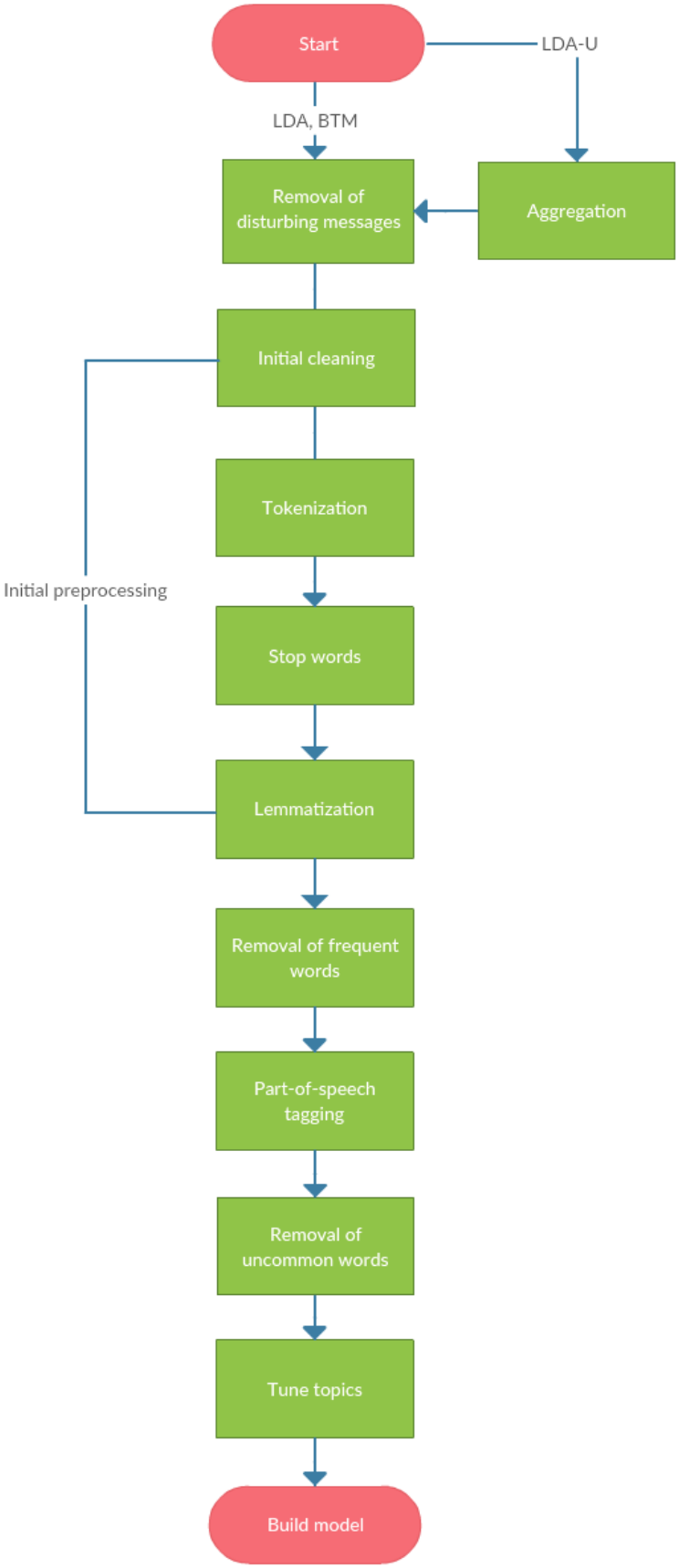


Figure 3.1: Flowchart showing the preprocessing steps.

3.1 Tools and dataset

For this work, we have used a Java implementation of LDA called MALLET (McCallum, 2002) and it uses collapsed Gibbs sampling when training a topic model. This is then run through Python using a wrapper provided by gensim (Řehůřek and Sojka, 2010). Our BTM code is taken from the original authors through their github page (Yan et al., 2013). This is mainly a C++ implementation with some parts in Python, which can be run through shell scripts. As for evaluating and visualizing our models, we have used several Python libraries, including spaCy (Honnibal and Johnson, 2015), gensim, pandas (McKinney, 2010), numpy (Oliphant, 2006), matplotlib (Hunter, 2007), pyLDAvis (Sievert and Shirley, 2014), scikit-learn (Pedregosa et al., 2011).

The dataset is a collection of 210,000 text messages in English. The majority of the messages have been used in the unsupervised machine learning algorithms, but we annotated 1000 of those messages for the classification step.

3.2 Aggregation

Since it is known that LDA does not work well on short texts (Yan et al., 2013), we took inspiration from Weng et al. (2010) by aggregating the messages from the same company. Aggregating means that we combine individual messages into larger documents. In our case, we chose to aggregate 10 individual messages. The new messages are referred to as pseudo-documents and are used as the input for a separate LDA model, which is further on referred to as LDA-U.

When it comes to classification, aggregating the messages leads to the assumption of that all messages in the same pseudo-document are related to the same topic. In reality, it might not be a relevant assumption, but if we are only interested in exploring the latent topics, it still serves this purpose.

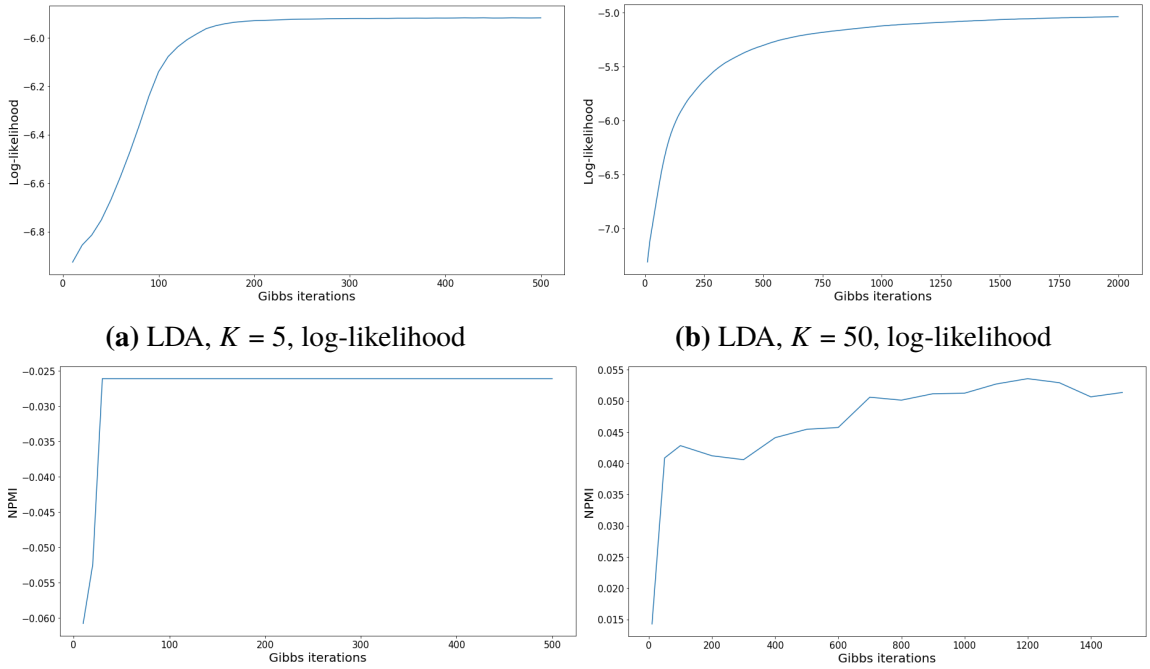
3.3 Hyperparameters

3.3.1 α, β

Before training our topic models, we had to set the two parameters α and β. The first idea would be to apply grid search, but due to BTM's large time complexity it was not an option. Instead, we used the recommended parameters from Yan et al. (2013), see Table 3.1. For the meanings of the hyperparameters, see Table 2.1.

Table 3.1: The set parameters for different models.

	LDA	BTM	LDA-U
α	0.05	$\frac{50}{K}$	$\frac{50}{K}$
β	0.01	0.01	0.01



(a) LDA, $K = 5$, log-likelihood

(b) LDA, $K = 50$, log-likelihood

(c) BTM, $K = 5$, NPMI

(d) BTM, $K = 50$, NPMI

Figure 3.2: Convergence for some of the models we trained.

3.3.2 Iterations

Prior to computing the number of topics, we had to assure the model converges for all parameter settings. By calculating the log-likelihood we could set the number of required Gibbs iterations. Since the model converges differently for different number of topics, we had to check the convergence of each model's log-likelihood. For example, Figures 3.2a-3.2b shows the approximated required number of Gibbs sampling iterations for LDA to converge for $K = 5$ and $K = 50$. Around 150 iterations are needed for $K = 5$, while around 1500 iterations are needed for $K = 50$.

For BTM, the log-likelihood is intractable. We would need to form the biterm multiset for the whole corpus, which approximately scales quadratically with the average length of the documents. Then, for each biterm, we need to get the probability of this biterm occurring in all the topics, sum them up and taking the logarithm. This process needs to be done for each model we want to analyze.

Instead, we can use topic coherence, which gives a score based on the word co-occurrences of the top words from each topic. This score can be computed reasonably fast and to some extent, also tells us how good the model is. Since we know that human evaluation and topic coherence are positively correlated (Chang et al., 2009), a higher score means a better model, and if the score plateaus at some level, this would mean the model is not getting much better.

As can be seen in Figure 3.2, the models requires different number of iterations for different values of K for convergence. In general, each set of parameters α, β, K requires us to check the convergence individually.

3.3.3 Number of topics

The number of topics K is a parameter we set before training our topic model algorithms. If we choose a too low number it can lead to multiple topics in a topic cluster, while choosing a too high number can lead to very fine-grained topics. We ran the topic models several times and recorded how their topic coherence varied with the number of topics. Using this, we could find a good value to start with.

We used two coherence score measures, namely NPMI and UMass. As can be seen in Figures 3.3-3.5, the actual topic coherence scores differ a lot. The explanation is that the scorers have different intervals. However, the important part is that the behavior of the graphs are similar. To identify the approximated number of topics for the dataset the "elbow" method was used. This means if an elbow shape can be identified in the graphs it has a potential of being the optimal or close to the optimal number of topics. In Figure 3.3 several elbows were identified. Both graphs for LDA strongly indicated $K = 10, 15, 25, 35, 40, 45$. Looking at pyLDAvis for these K we could discover $K = 10, 15$ was too few topics, because multiple topics could be found in a topic cluster. While for $K = 35, 40, 45$ duplicated topics could be found due to the topics being too fine-grained. Therefore, the initial value at 25 is a good value to continue with. The same reasons apply for both BTM and LDA-U, who had elbows at $K = 20, 25, 45$ and $K = 10, 15, 25, 30, 35, 40$ respectively.

Examining the topics closer, the best number of topics seemed to be $K = 25$. Note that the plots for BTM has a lower resolution. This is due to computational constraints.

3.4 Preprocessing

In the following parts, the preprocessing step is described in the same order as the preprocessing was performed.

3.4.1 Initial preprocessing

The dataset contains of messages with a lot of abbreviations, numbers, codes, links and non-alphabetic characters. The dataset also contains a lot of duplicated messages and messages following templates, where only a person name, code or time differs. In order to obtain high quality messages some preprocessing steps were required.

We started by removing all type of links, numbers, codes and non-alphabetic characters, because we assumed this type of information would not contribute to interpretable topics. When we tried to run the algorithms without removing these information, the topics were uninterpretable. Characters such as “www” and “_” appeared as top words. Note, in the example below the company name have been censored as “Anon#1”, but it was included when training the model.

Before: Anon#1 Rx: _NAME_, your Rx is due now. Reply REFILL to fill. Rx details: _URL_ HELP for more info & STOP to opt out of Rx alerts.
After: Rx your Rx is due now Reply REFILL to fill Rx details HELP for more info STOP to opt out of Rx alerts

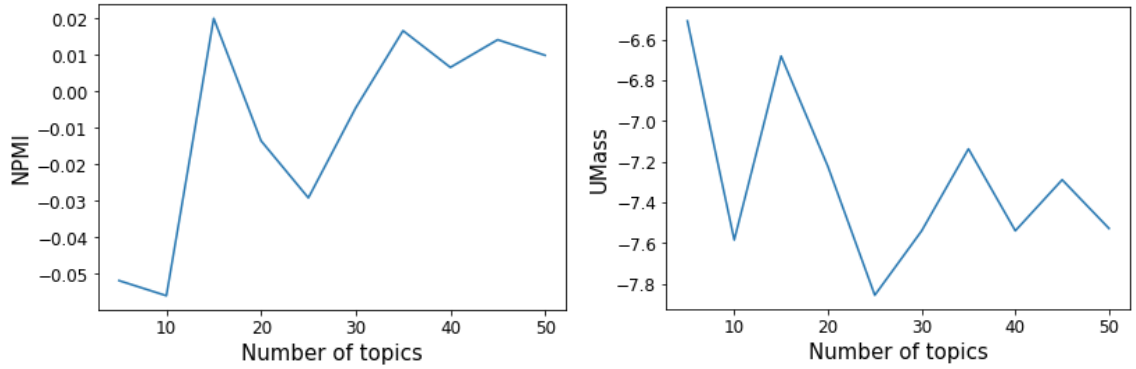


Figure 3.3: LDA topic coherence for different values of K. Note the elbows at $K = 10, 15, 25, 35, 40, 45$.

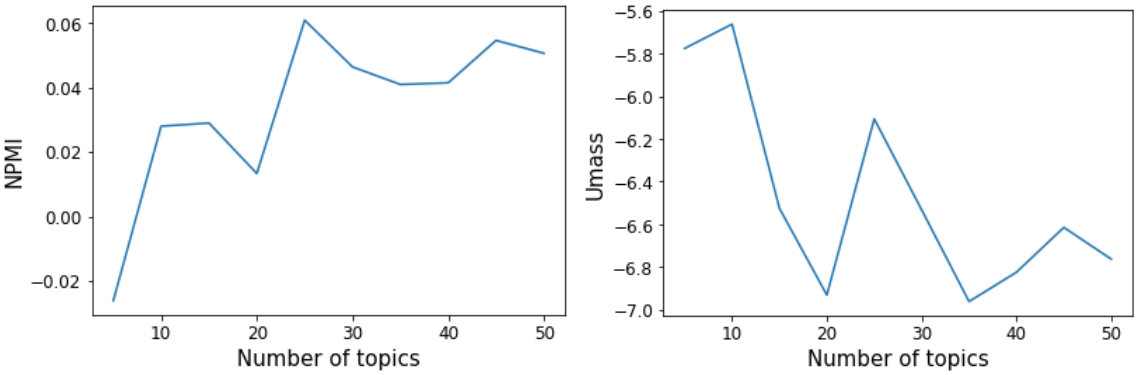


Figure 3.4: BTM topic coherence for different values of K. Note the elbows at $K = 20, 25, 45$.

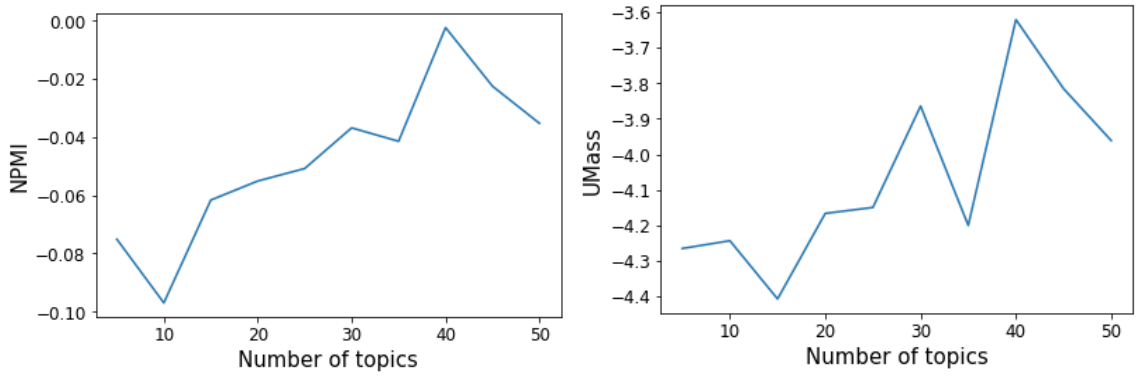


Figure 3.5: LDA-U topic coherence for different values of K. Note the elbows at $K = 10, 15, 25, 30, 35, 40$.

In the second step, all the messages were tokenized, lower cased and one letter words were removed. Tokenized means that the message is converted from a single string into a list of words. Note, in the example below the censored company name does not follow the preprocessing steps.

['Anon#1', rx', 'your', 'rx', 'is', 'due', 'now', 'reply', 'to', 'refill', 'fill', 'rx', details', 'help', 'for', 'more', 'info', 'stop', 'to', 'opt', 'out', 'of', 'alerts']

Grammatical words such as “your” and “for” are words that appear frequently, but do not contribute to informative topics. These words are called stop words. Filtering stop words is a common step in preprocessing text for various purposes. There are many different list of stop words for English. We chose the improved list from Stone, Denis, Kwantes (2010). This is also used in the Python libraries gensim and spaCy. This list contains 339 stop words.

['Anon#1', 'rx', 'rx', 'due', 'now', 'reply', 'refill', 'fill', 'rx', details', 'help', 'more', 'info', 'stop', 'opt', 'out', 'alerts']

Then we lemmatized the words, so “details” turned into “detail” and “alerts” into “alert”. The lemmatization step does also turn words such as “saw” into “see” and “mice” into “mouse”. This facilitated the text analysis, because it reduced the inflectional forms to a common base form. Stemming could also be used, but since stemming cuts the ending of a word the word could sometimes be difficult to interpret. For example, if stemming was used “applying” would turn into “apply” and “applies” into “appli”. Even though it works effectively for an English corpus, due to similar endings, the interpretation of the topic was assumed to be more important for our case, therefore we chose lemmatization. For our work, we used the lemmatizer from the spaCy library.

['Anon#1', 'rx', 'rx', 'due', 'now', 'reply', 'refill', 'fill', 'rx', **'detail'**, 'help', 'more', 'info', 'stop', 'opt', 'out', **'alert'**]

Lastly, documents that contained less than three words were removed. The motivation for this is the shorter the document is, the more difficult it will be to obtain a "correct" topic distribution of the document.

3.4.2 Removal of frequent words

In this step, we used the visualization tool pyLDAvis to examine the topics. As mentioned earlier, there are a lot of template messages in the dataset. To avoid clustering based on the templates more than the actual topics, we removed standard words such as “reply” and “stop”. These words appear among the top words due to being very common in the corpus, but do not contribute to a meaningful topic, see Figure 3.6.

In addition, removing these kinds of words not only makes the individual topics better, it also removes connections between topics that should not be there. For example, the word “stop” can be found in text messages regarding discounts and offers, but also in appointments as reminders of a meeting. In both cases, they represent the possibility to opt out of receiving any further messages of these kinds. However, the two kinds of messages belong to different topics, and they should not be clustered together. Since topic find each other depending on word co-occurrence in documents, removing these words also removes possible connections between topics. This results in less mixed topic clusters.

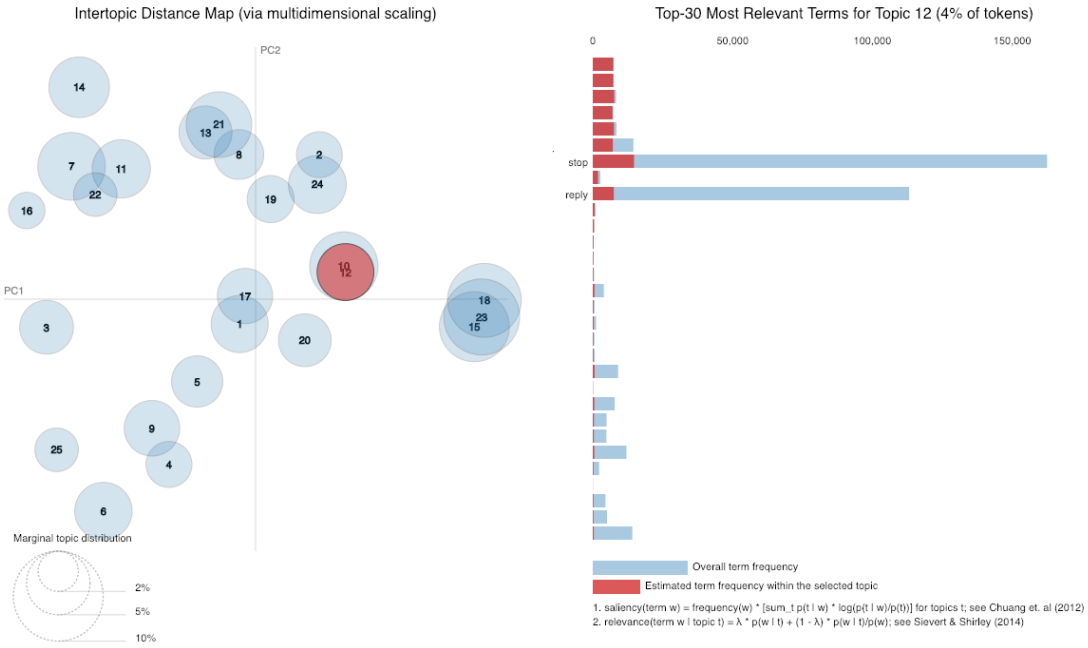


Figure 3.6: Illustration of removing standard text message words. In this case, we remove "stop" and "reply".

3.4.3 Collocations

The Python library gensim was used for detecting word collocations. When using the related functions we had to choose a scorer and a threshold. We chose NPMI as the scorer, because its score was limited to a specific interval of $[-1, 1]$.

Unfortunately, for Sinch's dataset word collocations did not have the same effect, because there are too many messages following a template. This resulted in many false word collocations. Therefore, we took the decision to exclude forming word collocations.

3.4.4 Part-of-speech tagging

POS tags stands for part-of-speech tags and are used to classify words depending on their meaning and usage. When performing the initial preprocessing steps, we kept all the words that were classified as nouns, adjectives, verbs, adverbs and proper nouns. This combination of POS tags will further on be referred to as the default combination. To analyze the POS tags, the topic coherence score was measured for different combinations. We then chose the combination with the highest score. The combinations of POS tags that we tested were default, only "nouns", "nouns and adjectives", and "nouns and verbs". In Table 3.2 we can see both NPMI and UMass recommends the default combination for LDA and BTM, while the "nouns and verbs" combination is recommended for LDA-U. NPMI and UMass do not agree on the best combination for LDA-U, but according to Röder et al. (2015), NPMI is better correlated with human interpretation. Note, the topic coherence scores varies a lot between the preprocessing steps. It is important to remember that the topic coherence value is an average value over all all topics and that there is no unit for this score.

Table 3.2: Topic coherence scores for different combinations of POS tags. Default refers to nouns, adjectives, verbs, adverbs and proper nouns.

	Default	Nouns	Nouns and adjectives	Nouns and verbs
LDA (NPMI)	**-0.042**	-0.120	-0.123	-0.086
LDA (UMass)	**-9.503**	-10.980	-11.213	-10.143
BTM (NPMI)	**0.019**	-0.076	-0.049	-0.017
BTM (UMass)	**-8.264**	-10.324	-10.019	-8.974
LDA-U (NPMI)	-0.045	-0.031	-0.024	**-0.018**
LDA-U (UMass)	**-4.981**	-6.357	-5.995	-5.204

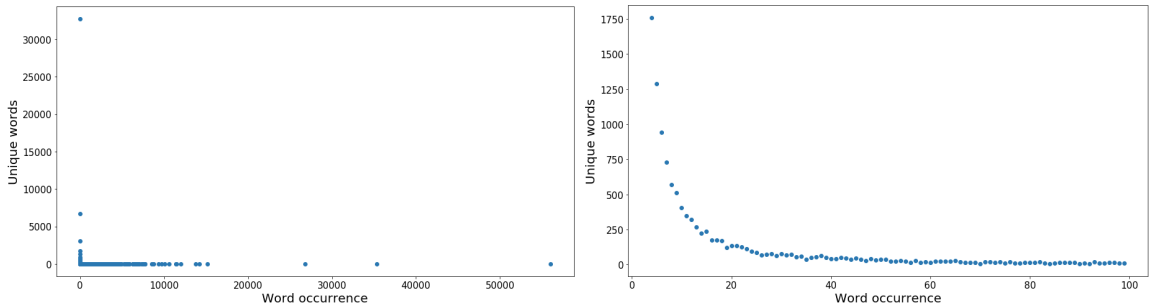


Figure 3.7: Plots over unique words with respect to their document occurrences.

3.4.5 Removal of uncommon words

Since we are working with business to customer messages, it is relevant to assume there are plenty of words only being used a few times. We thought of words such as person names. These words have a low probability of being elected as one of the top words due to Eqs. 2.8 and 2.16. To improve the model further, we removed words that do not appear frequently.

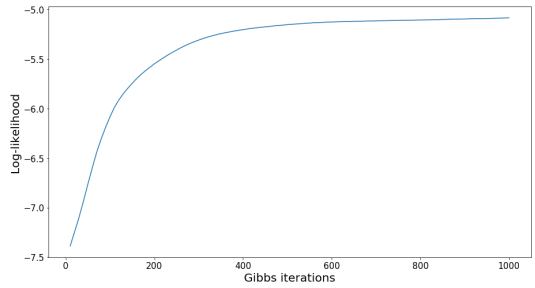
The question is how the threshold should be picked. We chose to determine the threshold by counting how many documents each unique word appeared in. By doing this, we could get an overview of the entire corpus. For example, we almost have 7,000 unique words appearing only in one document, and three words appearing in more than 20,000 documents, see Figure 3.7. The graph to the right indicates a cutoff around 10-20 occurrences will result in a smoother curve. By measuring the topic coherence score for the suggested thresholds, we could decide which one to use, see Table 3.3. Excluding the uncommon words will dramatically decrease the number of unique words and improve the model due to a better quality of the input.

3.4.6 Tune topics

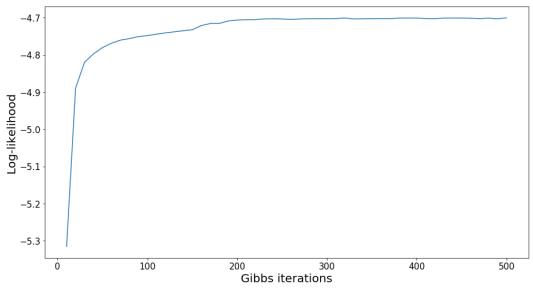
In the very last step, pyLDAvis was used to do some final fine tuning. For each topic we analyzed the top 10 words, because what the model found semantically close does not always correspond to the human interpretation as semantically close. The words we found irrelevant for the topic were added to the list of stop words.

Table 3.3: Topic coherence scores for different cutoff thresholds.

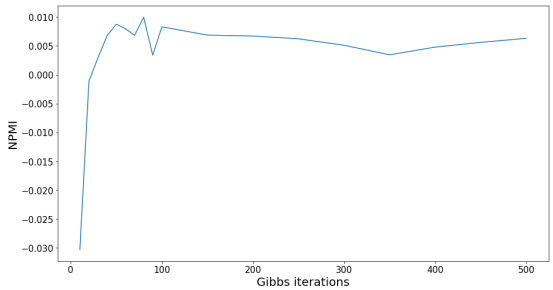
	10	15	20
LDA (NPMI)	-0.080	-0.085	**-0.045**
LDA (UMass)	-10.444	-10.336	**-9.455**
BTM (NPMI)	0.001	**0.030**	0.003
BTM (UMass)	-8.498	**-7.926**	-8.537
LDA-U (NPMI)	-0.057	**-0.020**	-0.026
LDA-U (UMass)	-5.737	**-5.201**	-5.332



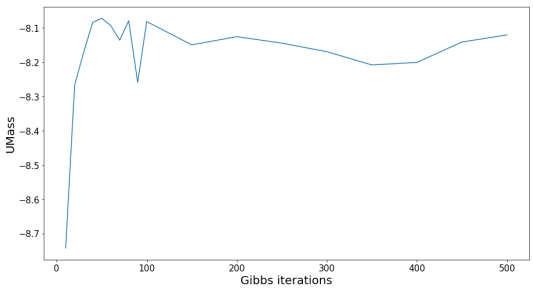
(a) Log-likelihood for final model of LDA.



(b) Log-likelihood for final model of LDA-U.



(c) NPMI for final model of BTM



(d) UMass for final model of BTM

Figure 3.8: Convergence for the final models we trained.

As we have mentioned, the dataset contains a lot of duplicated messages due to occasionally specific massive sendouts. The model could sometimes pick up on these messages, which made some topics less clear. We detected these disturbing messages and decided to remove them from the dataset.

On top of that, we also removed words which were part of some kind of template, as well as not contributing much to any topic. Examples of these would be some kind of code or unique identifier, or formal words to make a text message look more polite.

3.4.7 Convergence of chosen models

After all these preprocessing steps, we have finished choosing the parameters for preprocessing our dataset and hyperparameters for our models. Now, we need to ensure that we train these models long enough for them to converge. Therefore, we once again check the log-likelihood for LDA and LDA-U, as well as the topic coherence scores for BTM. These can be seen in Figure 3.8.

With these metrics ensuring the convergence of our final models, we decided to use

300 Gibbs iterations for BTM, 1000 for LDA and 500 for LDA-U.

3.4.8 Clustering documents

Both LDA and BTM can output a document-topic distribution. This distribution can be used to categorize the documents into different classes. For this, one can use the most dominant topic of each document, i.e. the topic which the document has the highest probability of belonging to. This way, one can cluster similar documents together as well as relate them to the words that dominate the topic they belong to. However, one needs to be careful when doing this, since documents which show a relatively high chance of belonging to several topics are less likely to be semantically related to their assigned class. Using document-topic distributions, we can also classify documents with K-means. We tried doing this, and it yielded very similar results to picking the dominant topic. The difference mostly lay in one topic which "absorbed" documents from all other topics. This result can be seen in Figure 3.9. Even then, the other topics in K-means clustering did not necessarily improve that much, while one topic was basically ruined. In addition, the F1-scores lay between 0.92 and 1 for all topics except for the predicted label 22 of Figure 3.9, which "absorbed" documents from the other topics. Therefore, we decided to move on using dominant topic as our classifying method.

3.4.9 pyLDAvis

The optimal value for relevance in order to identify topic was around 0.6, according to Sievert and Shirley (2014). This means that taking both local and global factors into account yielded the best result. This value seems to be specific for their own user study, however we also did some tests in comparing different values for λ, and we found that the value of 0.6 seemed to work fine. Therefore, we also chose to investigate our own topics using that value.

In addition, we mostly used relevance when analyzing and discovering topics, since pyLDAvis started sorting words by relevance instead of saliency whenever a topic was selected.

3.5 Human evaluation

Inspired by the steps in Chang et al. (2009), we designed two forms of human evaluation called word intrusion and topic intrusion. Because our goal of using topic models is to summarize the messages and produce semantically coherent topics, it is also important that humans can make sense of the topics. We test this by creating a survey based on our models, where word intrusion refers to picking a topic and represent it by a list of its top words. In addition, a word with a low probability within the same topic will be injected as an intruder. The task would then be for humans to correctly identify the intruding word. An easier example would then be the following words:

Angeles, Philadelphia, Atlanta, Los, Beijing, Washington

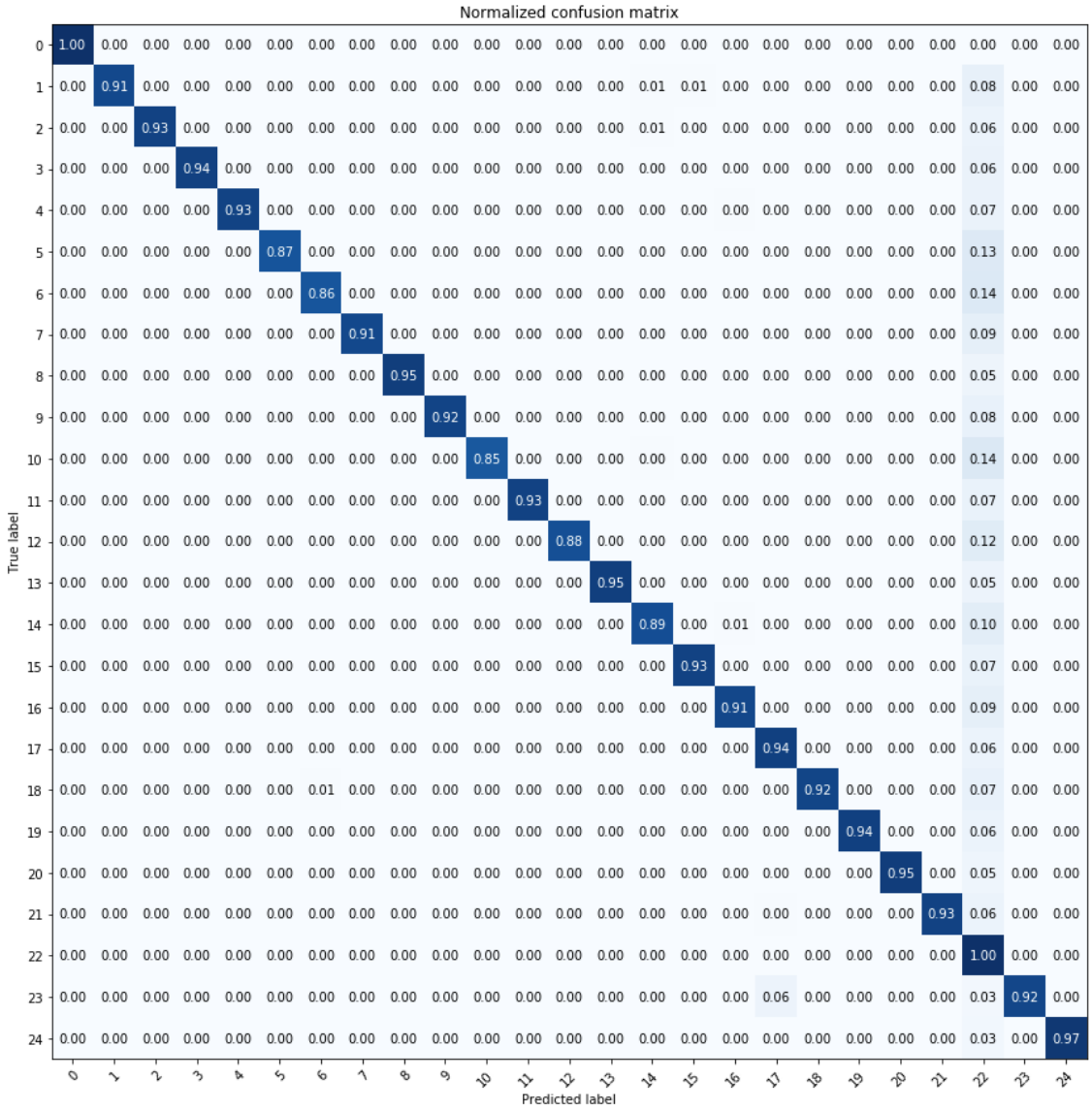


Figure 3.9: Normalized confusion matrix between dominant topic as true label and K-means as predicted label.

The intruding word could then reasonably be Beijing, as it does not lie in North America. Also note that the collocation *Los Angeles* has been split up into two alternatives. When creating the evaluation, we did not mention collocation, and purposefully left it to the people evaluating topics to notice it themselves. Since only one word could be the intruder, identifying a collocation excluded them from being the intruder.

Another example of word intrusion which is harder can look like this:

Tennis, Squash, Badminton, Soccer, Bandy, Golf

Here, it may not be so clear which word the intruder is. It could be *soccer* as it does not need equipment that acts as extensions of your body. However, considering the shape of the "ball" in these sports, the intruder could then be *badminton*, since it uses a shuttlecock which is far from a spherical shape. Other explanations could also justify other

alternatives. In our evaluation forms, we had word intrusions of several difficulty levels, ranging from the easy example to the hard example given. Even though the intruder of the harder example is ambiguous, there will only be one correct answer. Therefore, one can expect some questions to be hard and have a low human evaluation rate.

Topic intrusion, on the other hand, involves several topics as well as documents. Since it is possible to compute the probability of a document belonging to a specific topic, this is used in topic intrusion. By selecting a document and finding the document's top topics, i.e. the topics which the document is most likely to belong to, these topics will be represented by their top eight words. Once again, an intruder will be injected, chosen as a topic which the document has a low probability of belonging to. This topic will also be represented by its top eight words. With this setup, the task is to read the document in question, and then identify which topic is the intruder. If the document is too long, a few sentences at most will be presented instead of the whole document. An example of this would be the following sentence and topics:

Tour de France winner Lance Armstrong was accused of doping and had his awards stripped.

- *cancer, blood, doping, test, hormone, urine, performance, sports*
- *accuse, false, gossip, scandal, rumor, slander, talk, news*
- *election, candidate, party, government, poll, majority, result, minority*
- *cycling, road, gear, brake, bike, competition, saddle, tire*

In this case, one could argue that the third topic is the odd one out, since the example text mentions sentences and things related to scandals, as well as doping and cycling.

However, since we are working with short texts, we assume that each document only contains one topic. Therefore, we will do the topic intrusion differently. Instead of showing the three most dominant topics, we show only one dominant topic. We also show the bottom three topics instead of just one, i.e. we show the three least dominant topics for each document. An example of this would then be the following sentence and topics:

Table tennis is a sport.

- *ball, audience, arena, referee, winner, sport, court, intensive*
- *mechanics, photon, spin, molecule, quantum, quark, wave, particle*
- *chair, table, sofa, plastic, desk, wood, metal, fabric*
- *meat, cucumber, banana, vegetarian, pineapple, steak, potato, chives*

The sentence is related to sports, and the task would be to pick out the topic about sports, which is the first topic. The other topics can be quantum physics, furniture and food.

Since the dataset we worked with was based on text messages from companies, some words could have a different meaning or interpretation compared to a general situation. In

these cases, we chose to provide additional information, mentioning that it is some kind of company if it was a company, and writing out an abbreviation if it was an abbreviation. This information was given next to the word in parantheses. For example, the abbreviation *btwn* was written as *btwn (between)* and a company like *Sinch* was written as *Sinch (telecom company)*. Also, due to sensitive information, the survey was only handed out to employees within the company.

The examples given above can be seen as very easy to identify the intruder, except for the second word intruder example. Generally, the word intrusions might not be that easy, and even within one topic model, one topic might be better clustered than another. For this, we compute a coherence score for each topic. By making humans evaluate topics with high and low topic coherence, we can confirm the correlation between our chosen scores and possibly identify what factors make the topics clear.

We chose the topics uniformly based on their coherence score. For word intrusion in every model, we picked every other topic starting with the best one according to NPMI. If we reach the bottom topic, we restart from the second best and pick every other. This was done until we reached 15 topics. As for topic intrusion, we took 5 topics evenly spread out in topic coherence score. By doing this, we hoped to see if we could find any correlations between human evaluation and the coherence scores we had chosen. The choice of NPMI over UMass was due to Röder et al. (2015) showing better results for the former.

3.5.1 Text classification

So far, we have only explored the dataset without measuring any accuracy. In order to measure the accuracy of the document clustering performed in Section 3.4.8, we annotated 1,000 text messages that were not a part of building the model. The annotation and classification are based on the number of unique topics. The topics are named similarly to Table 2.3. We look at the top words and find a word that captures the meaning of the top words. In case that is not enough, we also check the top documents.

If multiple topic clusters are considered to have the same topic, they will be combined into one class. If one topic cluster consists of multiple topics, the top messages indicate which topic is the most dominant, and therefore which topic to choose. There is also a topic cluster named “others” for messages that do not fit into any of the detected categories.

When classifying the unseen documents, two different procedures are used for the models. For LDA and LDA-U, Gibbs sampling is used to infer the topic distributions. When Gibbs sampling was used for building the model, the topic assignments were randomly initialized. In this step, each word is instead assigned to the most probable topic according to the model. Since we have a trained model and a relevant initialization point, it is relevant to assume not many iterations are needed for the unseen documents to converge. We chose to set the number of iterations to 100. Note that, when inferring the topic distributions we do not affect any counts in the model. For BTM, Eq. 2.12 is computed when inferring the topic distribution.

It is important to mention topic modelling is not made for classification, but can be used with caution. We do not expect the classification to work excellently, because we are aware of that more topics can be found.

Chapter 4

Results

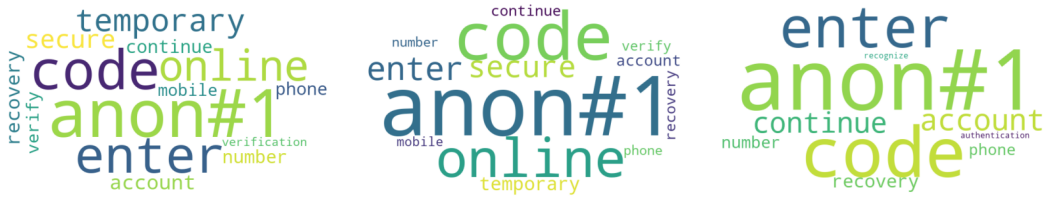
In this chapter, we present the results from the three topic models we trained. The first section presents a word cloud visualization of the found topics. The second section visualizes the topic clusters using pyLDAvis. The third section presents the identified topic names in descending order according to the topic coherence scores NPMI and UMass. In the fourth and fifth section the results from the human evaluation are presented and compared to the topic coherence scores. The last section presents the results from the classification.

4.1 Word clouds

The output from the topic models are topic clusters or more mathematically, 25 vectors with the probability of each word belonging to the topic. In the word clouds in Figures 4.1-4.3 we can observe three examples from the given dataset. All three models can identify a public transportation, verification code and job cluster. The words the models assume are semantically close are similar between the models, but the weight differs. A bigger font size indicates a higher weight. In Figure 4.3a and 4.3b we can observe that LDA's and BTM's job clusters attracted some bank respective housing related words.

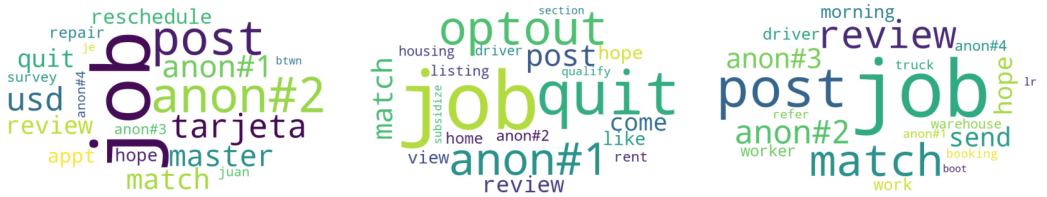
(a) Model: LDA **(b)** Model: BTM **(c)** Model: LDA-U

Figure 4.1: Word clouds showing the top words of a topic about public transportation.



(a) Model: LDA **(b)** Model: BTM **(c)** Model: LDA-U

Figure 4.2: Word clouds showing the top words of a topic about verification codes.



(a) Model: LDA **(b)** Model: BTM **(c)** Model: LDA-U

Figure 4.3: Word clouds showing the top words of a topic about jobs.

4.2 Topic coherence

In Tables 4.1-4.3 one can observe the topics in descending order according to the coherence score (NPMI). The coherence score gives us a quantitative measurement of how clear the topics are or how semantically related the words that form the topics are. The higher topic coherence score, the better the topic is. In Table 4.4, we have summarized the different topics each model could find. The three models could find similar topics, but there are minor variations.

4.3 pyLDAvis

In this section the distances between the topic clusters are visualized. Topic clusters that are semantically related will lie closer to each other, while clusters that are not related will be further apart. The distances are measured from the center of each circle.

Looking at Figure 4.4, five distinct groups can be found. The first group can be found in the upper right corner (6, 9). In Table 4.1, the reader can see these clusters were identified as topic clusters about verification codes. Moving downwards to another larger group of topic clusters (12, 13, 17, 20, 24) we can observe the majority of the bank balance and transaction related topic clusters have been collected. Nearby, to the left of this cluster other bank related topic clusters (3, 5) can be found. Below, we can find three topic clusters (8, 14, 23) about medical reminders. Lastly, in the left bottom corner the reader can find the topic clusters (2, 15) that are related to the banking support.

In Figure 4.5, we have the pyLDAvis presentation of BTM and four distinct topic cluster groups can be identified. In the upper left corner five topic clusters (3, 5, 9, 14, 18) about bank transactions can be found. To the right, we have three topic clusters (1, 2, 24) about

medical reminders. Lastly, in the upper right corner the reader can observe three topic clusters (16, 23, 25) about public transportation. The other topic clusters have gathered in the middle and are more difficult to distinguish.

The pyLDAvis visualization of the model LDA-U can be seen in Figure 4.6. The groups of topic clusters are more difficult to distinguish, but along the left side of the coordinate system six topic clusters (3, 6, 10, 18, 21, 23) related to bank transactions can be identified. To the right, topic clusters (7, 8, 19) about verification codes can be found. Note that topic cluster 19 has been given the topic name bank transactions in Table 4.3, but has a very close inter-distance to the verification code group. Reviewing Figure 4.4 and 4.5 again, we discover that bank transactions and verification codes clusters are in general close to each other in the coordinate system.

Table 4.1: LDA topics sorted by NPMI score. The topic names are labeled based on the top words and the corpus percentages are based on clustering using dominant topic.

Topic index	Topic name	Corpus percentage (%)	NPMI	UMass
19	Public transportation	2.9	0.288	-3.405
10	Public transportation	2.8	0.141	-5.067
9	Verification codes	3.6	0.109	-7.079
12	Bank balances & transactions	2.8	0.105	-5.590
7	Order status	2.8	0.099	-6.929
23	Medical reminders	2.8	0.094	-7.026
6	Verification codes	1.3	0.044	-8.013
20	Bank balances & transactions	3.7	0.030	-6.870
14	Medical reminders	3.7	0.008	-9.316
21	Services & logistics	4.0	-0.004	-9.601
24	Bank balances	5.0	-0.022	-10.029
17	Bank transactions	3.7	-0.049	-7.205
4	Mixed	4.7	-0.069	-5.492
1	Power outage	1.5	-0.069	-10.658
11	Tracking	2.7	-0.083	-11.257
18	Discounts & offers	4.4	-0.090	-7.854
13	Bank transactions	6.3	-0.117	-9.809
3	Bank notifications	4.0	-0.123	-9.570
8	Medical reminders	3.3	-0.146	-11.677
5	Bank loans & fraud detection	4.5	-0.166	-11.164
22	Jobs & appointments	4.1	-0.191	-12.439
16	Motivational texts	4.3	-0.214	-12.030
25	Reservations	2.1	-0.252	-14.316
15	Banking support	8.8	-0.311	-15.319
2	Banking support	10.1	-0.318	-15.773

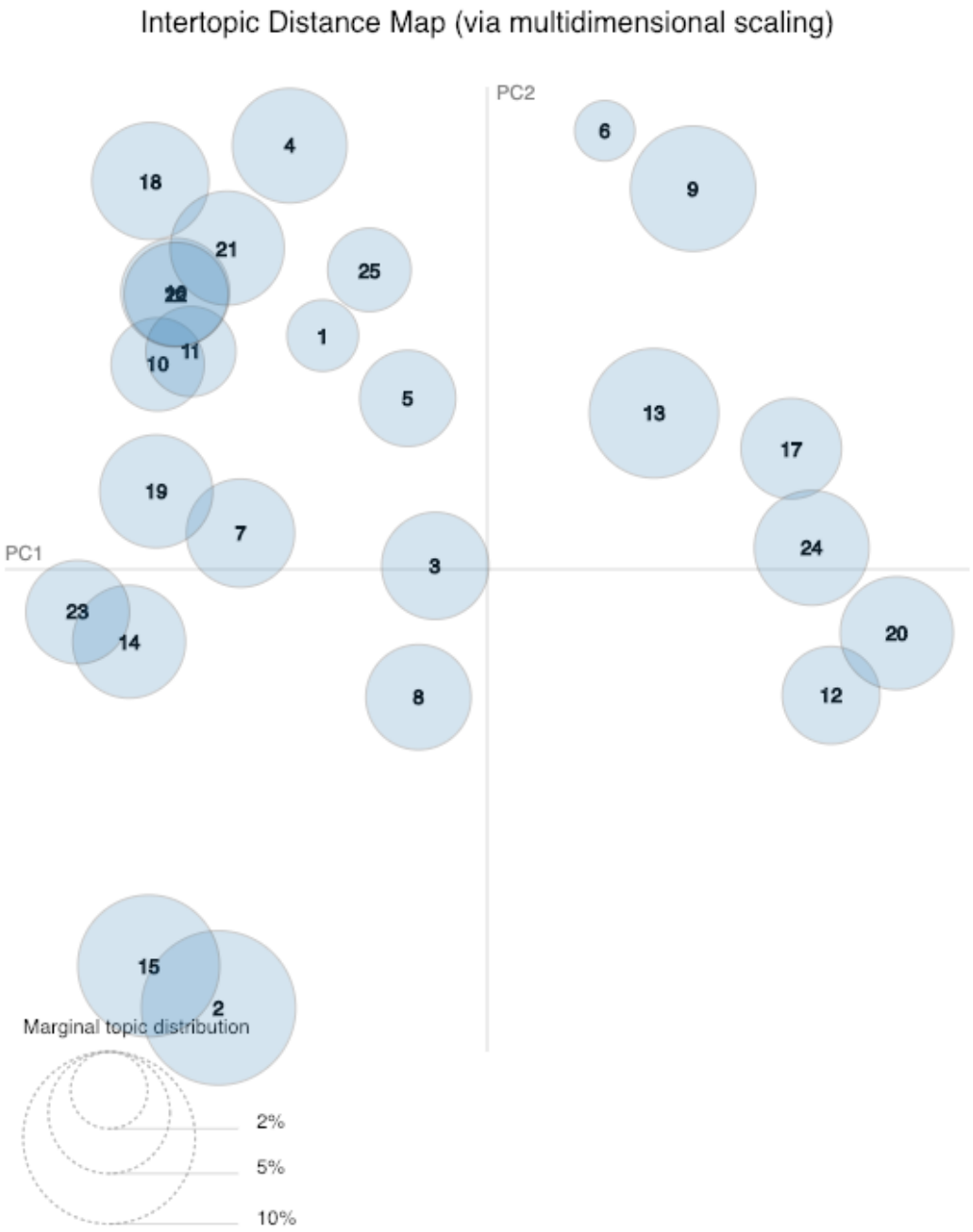


Figure 4.4: Dimension reduction of LDA model. The numbers refer to the "Topic index" of Table 4.1.

Table 4.2: BTM topics sorted by NPMI score. The topic names are labeled based on the top words and the corpus percentages are based on clustering using dominant topic.

Topic index	Topic name	Corpus (%)	NPMI	UMass
23	Public transportation	1.8	0.244	-3.442
25	Public transportation	2.3	0.177	-5.041
24	Medical reminders	2.9	0.145	-4.975
17	Verification codes	3.3	0.145	-6.654
9	Bank balances & transactions	2.9	0.139	-4.332
14	Bank balances	3.0	0.117	-4.703
16	Public transportation	2.2	0.095	-6.056
19	Discounts & offers	2.3	0.085	-7.055
5	Bank balances	5.0	0.034	-8.613
6	Mixed	8.4	0.028	-4.782
12	Flights	0.6	0.008	-10.569
10	Services & logistics	3.2	0.006	-9.737
1	Medical reminders	3.7	0.001	-9.322
7	Mixed	2.4	-0.020	-5.244
22	Appointments	3.3	-0.028	-8.258
20	Jobs	3.5	-0.038	-9.475
3	Bank transactions	3.9	-0.042	-7.436
18	Bank transactions	7.1	-0.045	-7.759
2	Medical reminders	4.3	-0.086	-8.487
8	Bank loans & fraud detection	4.8	-0.096	-11.180
13	Bank transactions	1.4	-0.124	-12.830
21	Banking support	18.8	-0.125	-11.178
11	Tracking	2.8	-0.137	-11.408
15	Bank notifications	3.8	-0.170	-12.933
4	Reservations	2.4	-0.185	-12.777

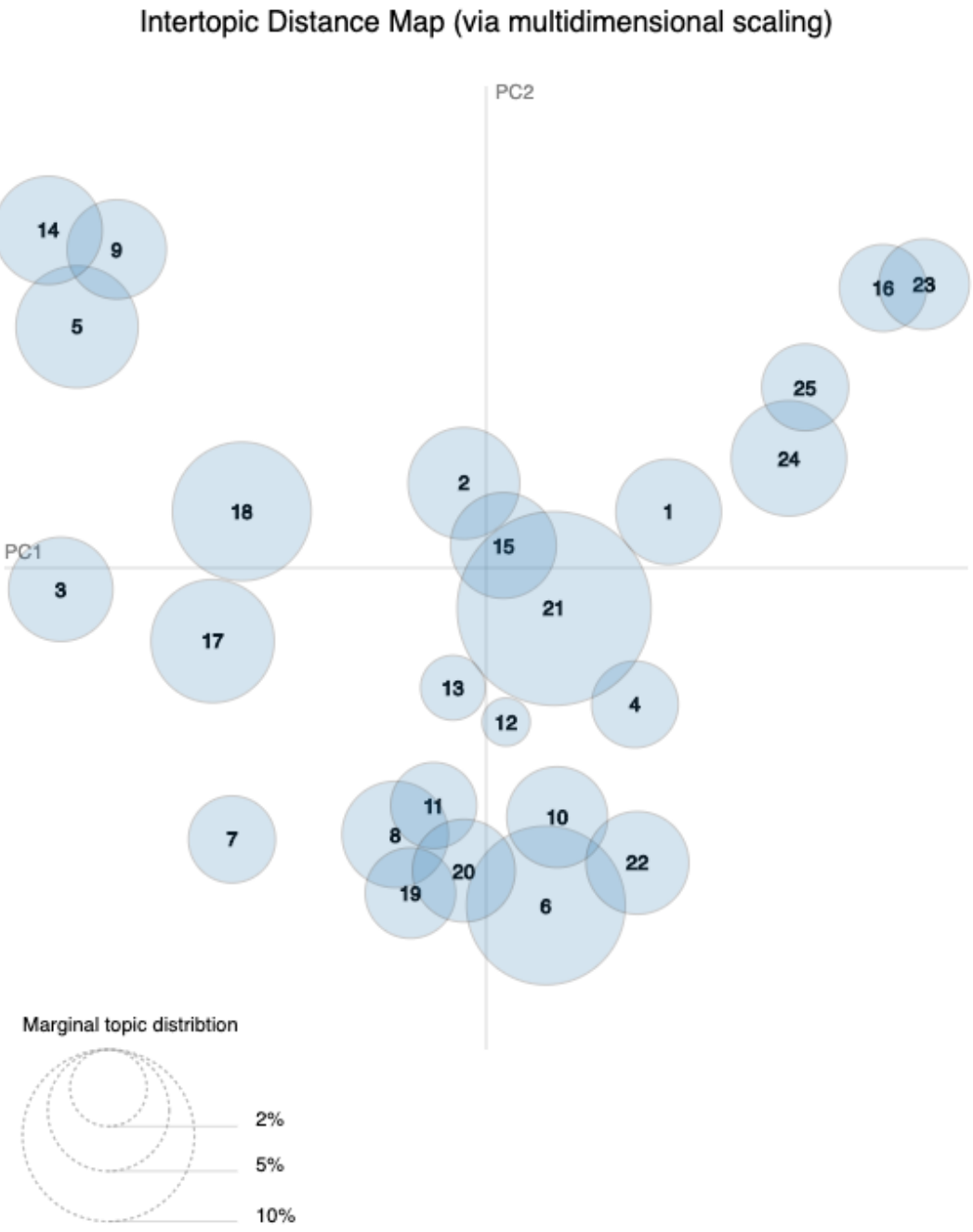


Figure 4.5: Dimension reduction of BTM model. The numbers refer to the "Topic index" of Table 4.2.

Table 4.3: LDA-U topics sorted by NPMI score. The topic names are labeled based on the top words and the corpus percentages are based on clustering using dominant topic.

Topic index	Topic name	Corpus percentage (%)	NPMI	UMass
24	Banking support	12.1	0.252	-2.827
5	Order status	2.4	0.248	-1.856
22	Medical reminders	4.2	0.171	-2.983
18	Bank balances & transactions	6.5	0.147	-0.982
10	Bank loans & transactions	4.8	0.120	-4.754
12	Services	3.6	0.116	-2.218
23	Bank balances	2.7	0.044	-5.102
3	Bank fraud detection	6.4	0.003	-4.389
9	Mixed	8.7	-0.013	-1.880
15	Public transportation	2.9	-0.027	-1.233
13	Public transportation	3.8	-0.040	-1.508
16	Medical reminders	3.3	-0.059	-6.994
20	Medical reminders	2.9	-0.061	-8.421
4	Mixed	2.0	-0.094	-11.174
7	Verification codes	2.2	-0.104	-5.612
6	Bank transactions	1.6	-0.105	-5.743
1	Jobs	4.1	-0.107	-3.178
11	Logistics & Power outage	2.3	-0.120	-6.121
14	Discounts & offers	2.9	-0.130	-4.934
19	Bank transactions	3.2	-0.130	-7.566
25	Bank notifications	3.7	-0.136	-8.639
21	Bank balances	3.8	-0.154	-9.291
17	Tracking	3.4	-0.197	-10.410
2	Appointments	3.6	-0.227	-9.046
8	Verification codes	3.0	-0.240	-13.178

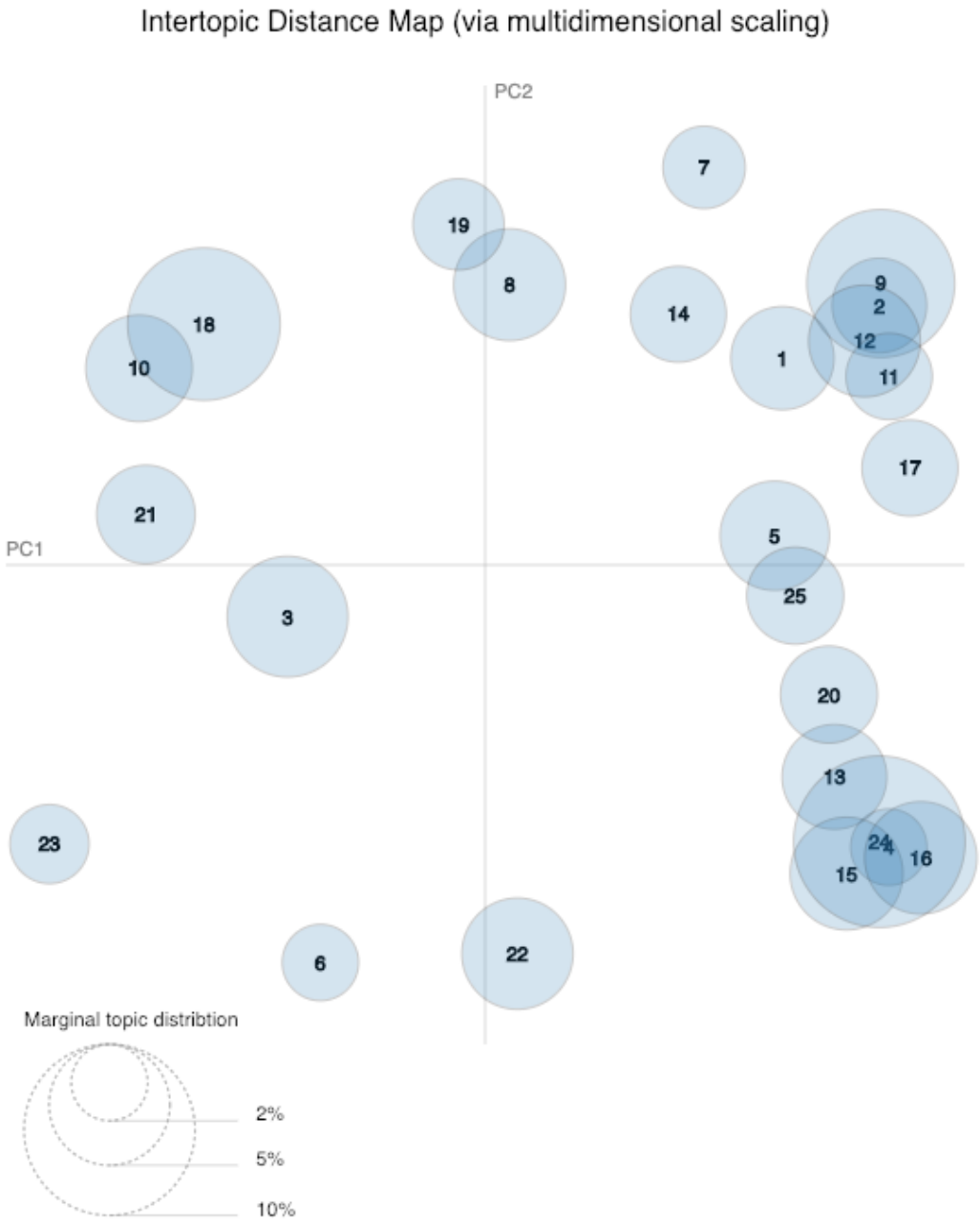


Figure 4.6: Dimension reduction of LDA-U model. The numbers refer to the "Topic index" of Table 4.3.

Table 4.4: Frequency and corpus percentage of topic clusters in different models.

	LDA		BTM		LDA-U	
	#	Corpus (%)	#	Corpus (%)	#	Corpus (%)
Public transportation	2	5.7	3	6.3	2	6.7
Verification codes	2	4.9	1	3.3	2	5.2
Banking	9	48.9	9	50.7	9	44.8
Order status	1	2.8	–	–	1	2.4
Medical reminders	3	9.9	3	10.9	3	10.4
Power outage	1	1.5	–	–	–	–
Power outage & logistics	–	–	–	–	1	2.3
Services	–	–	–	–	1	3.6
Services & logistics	1	4.0	1	3.2	–	–
Tracking	1	2.7	1	2.8	1	3.4
Discounts & offers	1	4.4	1	2.3	1	2.9
Jobs	–	–	1	3.5	1	4.1
Jobs & appointments	1	4.1	–	–	–	–
Appointments	–	–	1	3.3	1	3.6
Reservations	1	2.1	1	2.4	–	–
Flights	–	–	1	0.6	–	–
Motivational texts	1	4.3	–	–	–	–
Mixed	1	4.7	2	10.8	2	10.7

4.4 Human evaluation

In total, we got 33 responses. 9 of these were for LDA, 11 for BTM and 13 for LDA-U. Unfortunately, this sample size is not too big, but it is enough to give some indication of the trends we want to see. The survey also cannot be presented here, since it contains information that the company wants to keep secret.

4.4.1 Word intrusion

Following the process given in Section 3.5, we created and sent out surveys to get human feedback on our models. We can see the results of the word intrusion in Tables 4.5-4.7 with 15 word intrusions for each model. Each row in these tables represents one word intrusion, where the "Topic index" and "Topic name" are the same as in Tables 4.1-4.3. The percentages in the "Human evaluation (%)" column is the fraction that chose the correct answer. Also note that the tables are sorted in a descending order of NPMI score. We can see that the peaks of word intrusion lie around the fifth word intrusion and not in the top. In general, the top half scores higher than the bottom half.

4.4.2 Topic intrusion

Just like for word intrusion, the topic intrusion questions were created according to Section 3.5, with the same information given. Five topic intrusions were created for each model, and their topic index, topic name and the fraction of correct answers are given. The topics are sorted in descending order of NPMI score as well and the result can be seen in Table 4.8. Here, we see that there is a fairly strong correlation between topics and documents according to human evaluation. This means that the topic clusters summarize the semantic topics of the corpus well.

Table 4.5: LDA word intrusion results sorted by NPMI score. The human evaluation refers to the fraction that answered correctly in the survey.

Topic index	Topic name	Human evaluation (%)
19	Public transportation	44.4
10	Public transportation	66.7
9	Verification codes	66.7
12	Bank balances & transactions	77.8
7	Order status	77.8
6	Verification codes	44.4
14	Medical reminders	55.6
24	Bank balances	22.2
4	Mixed	33.3
11	Tracking	44.4
18	Discounts & offers	0
3	Bank notifications	22.2
25	Bank loans & fraud detection	55.6
16	Motivational texts	11.1
2	Banking support	44.4
Average		44.4

Table 4.6: BTM word intrusion results sorted by NPMI score. The human evaluation refers to the fraction that answered correctly in the survey.

Topic index	Topic name	Human evaluation (%)
23	Public transportation	27.3
25	Public transportation	9.1
24	Medical reminders	0
17	Verification codes	18.2
9	Bank balances & transactions	90.9
16	Public transportation	90.9
5	Bank balances	90.9
12	Flights	0
1	Medical reminders	54.5
22	Appointments	27.3
3	Bank transactions	45.5
2	Medical reminders	63.6
13	Bank transactions	9.1
11	Tracking	45.5
4	Reservations	45.5
Average		41.2

Table 4.7: LDA-U word intrusion results sorted by NPMI score. The human evaluation refers to the fraction that answered correctly in the survey.

Topic index	Topic name	Human evaluation (%)
24	Banking support	76.9
5	Order status	84.6
22	Medical reminders	61.5
16	Bank balances & transactions	92.3
10	Bank loans & transactions	100
23	Bank balances	100
9	Mixed	7.7
15	Public transportation	30.8
16	Medical reminders	84.6
7	Verification codes	46.2
1	Jobs	23.1
14	Discounts & offers	15.4
25	Bank notifications	69.2
17	Tracking	15.4
8	Verification codes	23.1
Average		55.4

Table 4.8: Topic intrusion for all models. The human evaluation refers to the fraction that answered correctly in the surveys.

Model	Topic index	Topic name	Human evaluation (%)
LDA	19	Public transportation	88.9
	21	Services & logistics	66.7
	13	Bank transactions	100
	8	Medical reminders	88.9
	25	Reservations	55.6
Average			80
BTM	23	Public transportation	100
	14	Bank balances	100
	22	Appointments	100
	20	Jobs	100
	21	Banking support	72.7
Average			94.5
LDA-U	18	Bank balances & transactions	100
	23	Bank balances	84.6
	13	Public transportation	69.2
	1	Jobs	100
	2	Appointments	84.6
Average			87.7

4.5 Classification

Using the trained models, we can infer the document-topic distribution of new unseen documents as well. After annotating previously unseen messages manually, we can compare this label with the dominant topic of each model prediction for the annotated messages. These comparisons can be seen in Figures 4.7-4.9, as well as the F1-scores for each topic in Table 4.10. Since the confusion matrices are normalized, we also need the information of Table 4.9, which states for each model, how many documents belong to that cluster.

For example, in Figure 4.7, we can see that the model correctly classified all 7 messages within "Power outage", but it also misclassified 1% of "Verification codes", 8% of "Reservations" and 2% of "Others" as "Power outage" messages as well. Converting this into numbers, this means 1 "Verification codes", 3 "Reservations" as well as 2 "Others" messages were misclassified. This means that even though the normalized confusion matrix looks good for classifying "Power outage", it does not tell the full story by itself. Looking at the F1-score of LDA's "Power outage" class, it drops by quite a bit. The same can be said for LDA's "Motivational texts"; one has to use the confusion matrices in conjunction with the tables below.

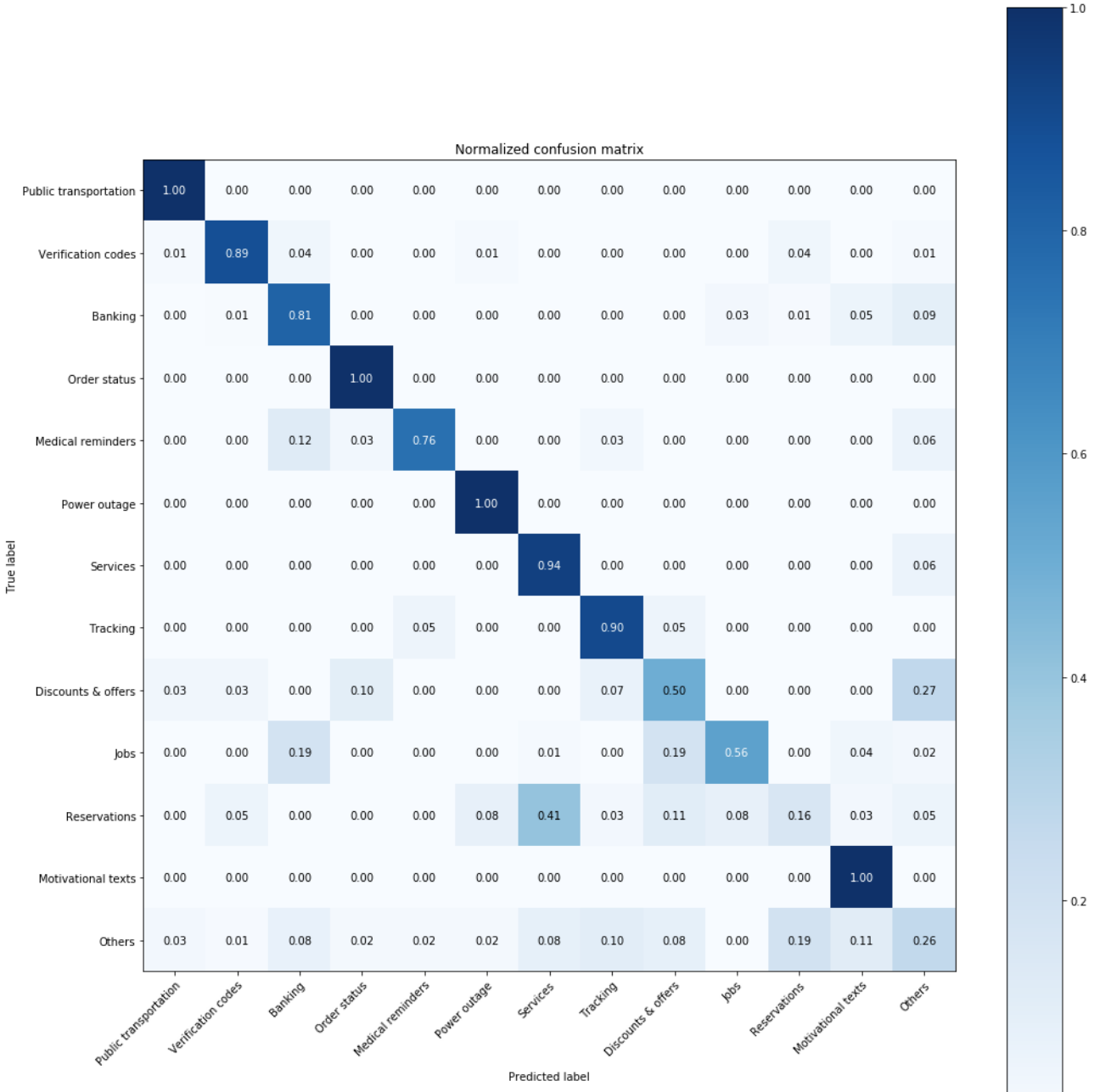


Figure 4.7: Confusion matrix from the final LDA model on an annotated test set.

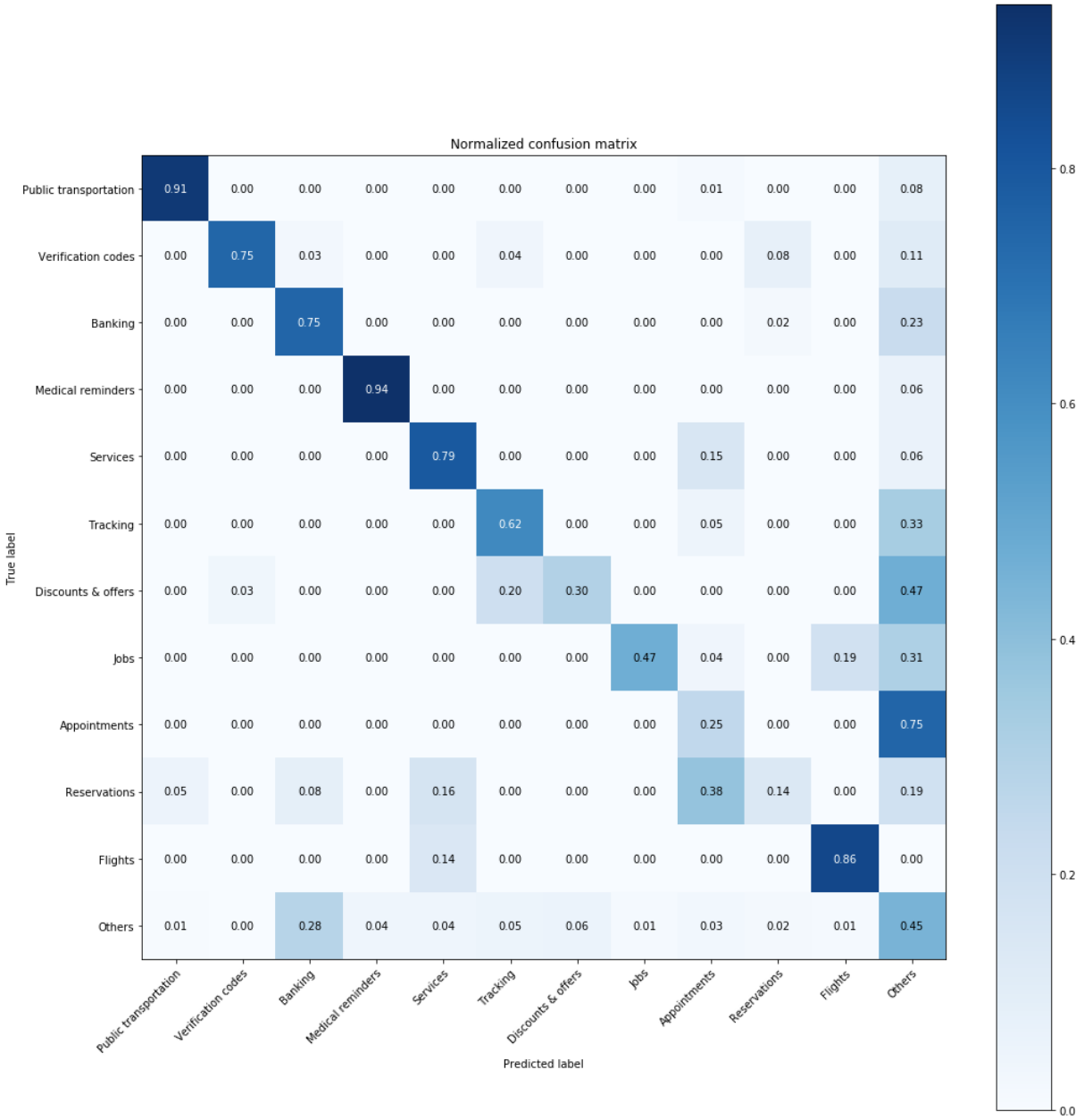


Figure 4.8: Confusion matrix from the final BTM model on an annotated test set.

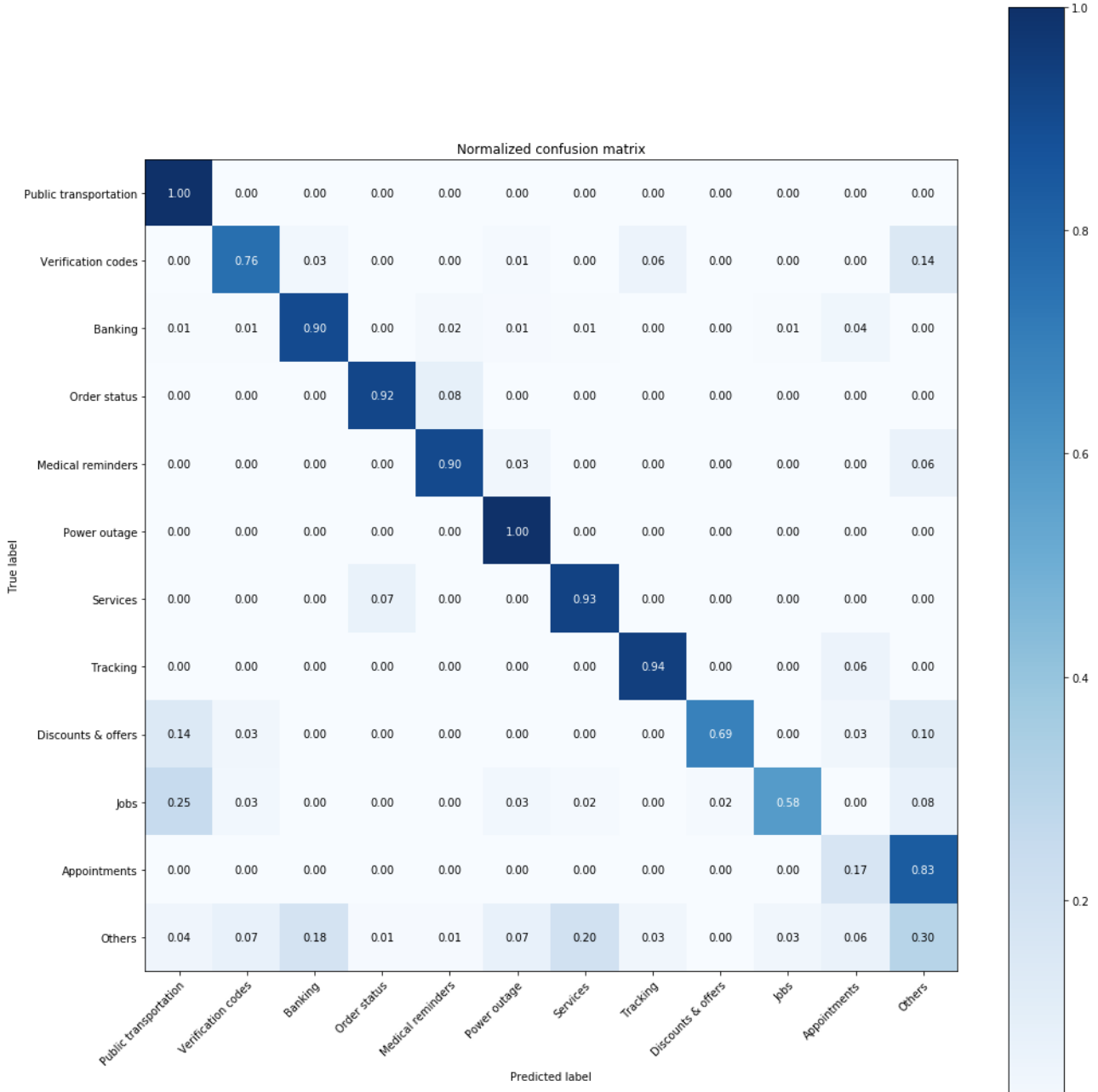


Figure 4.9: Confusion matrix from the final LDA-U model an annotated test set.

	LDA	BTM	LDA-U
Public transportation	87	87	87
Verification codes	79	79	79
Banking	440	440	440
Order status	30	–	30
Medical reminders	33	33	33
Power outage	7	–	7
Services	34	34	34
Tracking	21	21	21
Discounts & offers	30	30	30
Jobs	81	81	81
Appointments	–	8	8
Reservations	37	37	–
Flights	–	7	–
Motivational texts	3	–	–
Others	118	143	150

Table 4.9: Number of annotated documents in each class.

	LDA	BTM	LDA-U
Public transportation	0.97	0.93	0.85
Verification codes	0.90	0.85	0.78
Banking	0.86	0.81	0.92
Order status	0.91	–	0.90
Medical reminders	0.81	0.87	0.79
Power outage	0.70	–	0.41
Services	0.70	0.73	0.64
Tracking	0.68	0.51	0.77
Discounts & offers	0.40	0.38	0.80
Jobs	0.64	0.62	0.69
Appointments	–	0.11	0.06
Reservations	0.16	0.17	–
Flights	–	0.40	–
Motivational texts	0.13	–	–
Others	0.30	0.33	0.40
Average	0.63	0.56	0.67

Table 4.10: F1-scores for each model and class.

Chapter 5
Discussion

5.1 Dataset size

Even though we had 210,000 messages available, having more data could have given us more information. If we had gotten datasets from several time periods, we could have found trends which are time-dependent by separately training models on data from certain time periods. Some topics could have an increasing amount of messages sent as time passes. New topics can also appear that did not appear before. Conversely, some topics may decrease in size, meaning that less people use text messages for some purpose. In extreme cases, those topics might even disappear.

5.2 Hyperparameters

Number of topics. In our results, we have some clusters that are mixed in their content, containing two or more semantic topics. During our experiments, we found that increasing the number of topic clusters does not always equate to more semantic topics. Instead, there were cases where a semantic topic split itself up. An example of this would be the semantic topic banking, which was split into topic clusters such as bank transactions, bank notifications, etc. An explanation of why this happens is that the same semantic topic can use different words to convey the same meaning. Since the models work based on word co-occurrences, this means that topics will not cluster together depending on the word choice in the documents. For example, in two of our bank transaction clusters, one of them used abbreviations while the other used the full word. With our dataset, we found 25 topics to be a good trade-off between coherent topics and getting as many unique topics as possible. Ultimately, this yielded 15 unique topics during our own evaluation.

Another thing to note when increasing the number of topics K is that the topic coherence score will vary and find local maxima. This can be seen in Figures 3.3-3.5. Higher

values of K correspond to more fine-grained clustering of the dataset. For Sinch's dataset, it would however result in more company specific topic clusters and less about topic clusters that make sense semantically.

Convergence. The convergence for Figure 3.2c compared to Figure 3.2d is interesting. In the case of Figure 3.2c, the coherence score converges very fast due to the top 20 relevant words remaining the same. After this seeming stabilization, we do not know anything about the words below top 20. These words may still vary in their probabilities. We are still satisfied with such a result, since we judge the topics mostly by their top words. When we have a higher number of topics which we can see in Figure 3.2d, the words that come from the frequent topics in the corpus can spread out over the different topics. Since they are not enough to "fill" all topic clusters, we can see how the other (less frequent) words still keep changing their topic assignments in the fluctuations of the graphs.

α and β. Regarding the other hyperparameters α and β, we can reason about whether our set values make sense. For β, we only want a fraction of all the words to be frequent within a topic. Consequently, we expect a value of $\beta < 1$ to be reasonable, since it is the parameter of a symmetric Dirichlet distribution. We used the same value as the authors of BTM, setting $\beta = 0.01$. As for α, the default value of $50/K$ given by the authors for BTM performed satisfactorily as well. Having $\alpha > 1$ seems reasonable for BTM, since we can expect a fair amount of every topic in the corpus.

The same value of $\alpha = 50/K$ was used for LDA-U, and its results were reasonable. Using $K = 25$, we got $\alpha = 2$, which means that each document is likely to contain several topics. It works for LDA-U, since it uses aggregated pseudo-documents which can contain multiple topics. This value of $\alpha = 50/K$ also seemed to be fairly common when we did our literature study.

Meanwhile for LDA, one of the problems we ran into was that using $\alpha = 50/K$ resulted in very even document-topic distributions. In short texts, this is not desirable, since we want roughly one dominant topic in each document. We expected a lower value to work good then, at least $\alpha < 1$, due to the same reasons as $\beta < 1$. We tried some values which others have used, e.g. $\alpha = 0.1$ and $\alpha = 0.05$. Trying these, we evaluated the models ourselves and found $\alpha = 0.05$ as the best value.

Grid search. One might wonder why we did not use any form of grid search when optimizing our hyperparameters. The biggest problem in our case was the time constraint. We wanted a fair comparison between the models in terms of the models having converged towards the actual distributions. For each set of hyperparameters, we would have to train one model. Considering that we had many combinations of hyperparameters to try out, this would result in requiring to train many models. However, BTM took a very long time to train. Therefore, doing a grid search to find the optimal hyperparameters was not a viable option.

5.3 Preprocessing

In this work, many preprocessing steps were used to obtain the high quality input messages. Obviously, there are steps that could be improved. For example, the disturbing messages that were detected through the results of the topic model output. If the time of sending the messages had been accessible, a filter could have been implemented to remove the massive mailings of identical messages during a short time interval. Implementing this filter could have led to clearer topics, because we noticed several disturbing messages became top words, which made the topic difficult to interpret.

Another preprocessing step that could be improved is the initial cleaning. It was challenging to clean a dataset with a lot of abbreviations and non-alphabetic characters. In the beginning, we tried to split words with non-alphanumeric characters as delimiters. This led to an overabundance in abbreviations and topics that were impossible to interpret. Therefore, we came to the conclusion of removing all words containing non-alphanumeric characters. This led to much clearer topics, but the downside was the loss of interesting information and topics. The improvement could be to find a way to extract the important information from parts containing non-alphanumeric characters and a way to convert the abbreviations into a common base form, or at least into human interpretable words.

When we explored other datasets, finding collocations was a step that contributed to a better understanding of the topics. In this dataset, it was not possible to apply this step due to many messages following predefined templates, which led to "fake" word collocations. Since the dataset consists of a lot of company names and other word combinations that could improve the interpretation of the topics it would be interesting to search for an alternate way to extract only the real word collocations.

5.4 Topic coherence

In Tables 4.1-4.3, the topics have been ranked in decreasing order according to NPMI. The order would be different if it was sorted according to UMass, but the most important is to see whether the general trend is the same. The top and bottom topics should generally be the same. Note that topics such as “Jobs”, “Appointments”, “Reservations” and “Tracking” are generally found among the bottom topics.

The topic coherence scores for the topics in LDA-U differed quite much compared to the other models. For example, “Banking support” was among the bottom topics for both LDA and BTM, but for LDA-U it was the best topic, while the opposite goes for “Verification codes”. This result was very unexpected and we cannot explain why, besides suspecting that it must be related to the aggregation.

5.5 Corpus percentage

In Table 4.4, we can find the corpus percentage of topic clusters in different models. The models mostly agree with each other, but there are some interesting parts that differ. The first is that LDA found one more topic than BTM and LDA-U. This might be the explanation of why the mixed corpus percentage is much lower than the other models. Secondly,

BTM managed to find a small cluster of flight messages and LDA managed to find a cluster of motivational texts. We assume this is due to the randomized initialization of the algorithms. Finally, according to the presented results, BTM is the only model that could not find anything related to power outage. This is because it was in a mixed topic together with loans and fraud detection. We chose to mark the topic as "Fraud detection" (topic 8) in Table 4.2.

Note that one has to be careful with the corpus percentages given in Tables 4.1-4.3. Infrequent topics that do not occur often in the corpus can gather in topic clusters with a frequently occurring topic. In these cases, such words might not appear among the top words, but they still affect the corpus percentage. For example, we know from the BTM model that there is a "Flight" topic, but it does not appear in LDA or in LDA-U. This cluster is most likely hidden in another cluster, maybe in public transportation. Situations like these make the corpus percentage column of Tables 4.1-4.3 unreliable to read off directly, but they give a hint of what is correct.

5.6 Visualization

Both the topic modelling and the visualization through pyLDAvis are forms of dimension reduction. The original corpus in bag-of-words form can be seen as a $D \times V$ matrix, with each row representing a document and each column representing a unique word. An element would be the number of occurrences of each unique word in a document. After the dimension reductions from our topic models and then pyLDAvis, we reach the results of Figures 4.4-4.6. In these figures, we can still find similarities between models. The topics that are semantically similar generally lie close to each other. Some of the clusters that distance themselves from the large clump also belong to the same topic, which is banking in general. This indicates that the different models find similar patterns in the dataset. This is also further verified by looking at the word clouds of Figures 4.1-4.3.

As mentioned in the theory chapter, we know that our pyLDAvis figures use PCA for dimension reduction. As we used a library to do this dimension reduction for us, some intermediary steps are lost. One such piece of information that is interesting is the variance along each principal component, which corresponds to how different the documents are along an axis. In other words, finding out the variance of each axis would tell us how well the pyLDAvis figures represent the document-topic distribution. If there are three major principal components and the rest are minor, then we would know that a 3D-plot could be suitable for visualizing the topic distributions of each model.

5.7 Human evaluation

Word intrusion. In the results for word intrusion (see Tables 4.5-4.7), it is hard to determine whether the NPMI scores are related to human evaluation just based on the tables. Talking to people that had completed the survey, the general impression was that the intruders were hard to figure out. This can be seen in the results as well.

However, the cause for difficulties can differ from one word intrusion to another. During the creation of the survey, we decided to pick a bottom word from each topic, and then

accept it as long as we could make sense out of it. Generally, bottom words (i.e. intruders) do not appear together with the top words in documents. Still, they can make sense together with the top words in the survey. An example of this is the word “between”, that appeared in a topic about public transportation. In this case, this word can be interpreted as some form of connection, either between locations or in a time period. Since the top words of this cluster included the collocation “exact fare” (split up in two alternatives), it is reasonable to think of “exact” as the intruder instead of “between”, if one does not recognize the collocation.

In other cases, the topic clusters may be mixed and therefore have words from two or more semantic topics among the top words. Adding an intruder results in even more confusion. The intruder could be combined with some of the top words to create a topic which does not exist in the corpus. This happened in the “Flights” cluster of Table 4.6. The other topic among the top words was related to sensors, while the intruder was “urban”. We then suddenly have a topic about sensors in urban areas, and people start guessing wrong.

The intruder might also coincidentally form a topic according to humans, even though the corpus did not include this word co-occurrence. One has to remember that the basis for LDA and BTM lie in word co-occurrence in documents. If an intruder belongs to the same topic according to humans, all words suddenly become candidates for being the intruder. The end result would then depend on what words people think belong "more together". This happened for example in “Discounts & offers” in Table 4.5. The words in this word intrusion were:

confirm, appointment, redeem, kiosk, respond, accept

The last word *accept* was the intruder, while the choice of humans was *kiosk*.

There is an important difference when comparing the circumstances for our human evaluation and the intended receiver. The intended receiver may have additional background knowledge about the message in some cases. It can be that the receiver decided to subscribe to a certain type of messages, and the message can use abbreviations which are specific to the topic or the person’s circumstances. Therefore, if the dataset had less templates and the language varied more, the word intrusion may have yielded better results.

When analyzing the results in Tables 4.5-4.7, we can see that all models have a peak, not in the beginning, but closer to the fifth highest word intrusion. This might have been a coincidence, but it could also have to do with people learning how word intrusion works as they do it. The order which the word intrusions were presented for each model was basically in order of NPMI score, except for the second and fourth topic from the top, which appeared last. This, together with the problems discussed above, can explain why the top topics were not as correlated to human evaluation as we had expected. What we should have done is to give them some warm-up questions before doing the actual survey. Taking this into consideration, we can see hints of a correlation between human evaluation and topic coherence: The higher topic coherence, the easier it was to answer correctly, except for the problems discussed above about word intrusion.

The overall performance among the models was comparable to each other. Judging from the average correct answers, LDA-U was slightly better.

Topic intrusion. Since topic models have topics as outputs, we also have to make sure that these are closely related to the actual perceived documents. That is why we created the topic intrusions. Looking at the results in Table 4.8, we can see that people generally could find the correct topic. In other words, this shows that there has to be similarities between the top topic and the document given in a topic intrusion. This relation justifies using topics as a summary of a document.

Considering the overall results, BTM performed the best in topic intrusion, which we can see in Table 4.8.

5.8 Classification

What would be interesting to discuss is how the classification went for the topics that were considered as a mixture of two topics. In Table 4.4, we have 15 unique topics and three of them are considered as a mixture of two topics. We have "Power outage & logistics", "Services & logistics" and "Jobs & appointments".

Starting with the topic "Power outage & logistics", we can see that LDA was the only model that could identify a clean cluster about power outages, while LDA-U found the mixed topic. Looking at the F1-scores in Table 4.10, we see that LDA has a much higher F1-score than LDA-U, just as expected. In the confusion matrices found in Tables 4.7 and 4.9, we can see that all messages labeled as this topic are correctly classified, but there are other messages that are not labeled as this topic that have been classified as "power outage". This tells us the classification agrees with the human evaluation that the topic is a mixture of multiple topics.

Doing the same analysis with "Services & logistics", the majority of the messages have been correctly classified. Looking at the confusion matrices in Tables 4.7-4.9, the misclassified messages are from "Reservations" and "Others". This was anticipated, due to service related reservations and that the logistics messages have been labeled as others. Note that LDA-U has a lower F1-score compared to other models, even though it was considered as a clean topic. The explanation of this is, when naming the topics, we looked at samples of the documents. If we had sampled more documents, we could have discovered a better representative for this topic.

For "Jobs & appointments", it was only the LDA model that did not manage to separate the topics. Interestingly, the F1-score for "Jobs" is around the same for all models. Looking at the confusion matrices, approximately half of the messages labeled as "Jobs" were correctly classified. When annotating it was relatively easy to identify the job messages, so the problem most likely lies in the models having issues with distinguishing between related topics. Another issue could also be in the preprocessing step due to removal of too much information.

The conclusion from these three topics is that the human evaluation and the classification correlates in the sense of that these topics are a mixture of multiple topics.

In Table 4.10, "Appointments", "Reservations" and "Motivational texts" are the topics with very low F1-score. This is it not surprising, because all three topics are among the bottom topics according to the topic coherence scores in Tables 4.1-4.3. Another interesting thing to note in Table 4.10 with all the F1-scores is that it partly agrees with the topic coherence scores. They agree with each other on which topics are the clear ones and

which topics are the unclear ones. The conclusion we can draw from this is that the topic coherence scores are relevant.

In “Discounts & offers”, LDA-U has a much higher F1-score when compared to the other models. It is not obvious why, because the topic coherence scores nor the human evaluation can explain the large difference. The only possible explanation is that the aggregation helped. Since there is a larger variation among the templated messages in “Discounts & offers”, it is relevant to assume the number of unique words is larger in the aggregated messages. This facilitated the model to find the connection between different words.

Overall, these problems all assume that the annotated data is correct. In our case, we allowed one person to annotate all 1000 messages. One upside is that similar messages will be consistently classified the same, however if there are messages that are hard to put a label on, the downside is that we do not know whether those messages are correctly classified. On the contrary, those messages might all be wrongly classified. It would have been better if several people annotated the same messages. This way, if someone is unsure of how to label a message, others can help. Annotating like this would improve the "hard to classify" messages.

To improve the classification, we could have balanced the uneven distribution of classes in Table 4.9. As can be seen, there are a lot of messages related to “Banking” and a very few related to “Flights” and “Power outage”. For the latter, it is difficult to say whether they are representative for all the messages in that class. Therefore, their scores in Figures 4.7-4.9 and Table 4.10 also become less reliable.

Judging the models strictly based on F1-scores, we can see that their performance are roughly in the same class, but LDA-U is the best performer.

5.9 Expectations

When we explored the topic models with news headlines there was no doubt BTM discovered more and better topics than LDA. Therefore, the expectation we had was that BTM would outperform LDA on Sinch’s dataset as well. According to the results obtained there are no clear signs of that BTM works noticeably better than the other models for Sinch’s dataset.

The interesting question is: Did BTM work less good or did LDA work better on this dataset? The largest difference between the news headlines and Sinch’s dataset is that the latter consisted of many template messages. We know that LDA has sparsity issues when it comes to short texts, but somehow it performed equally good as BTM. We assume this is due to the many template messages, which helped the model to find the connection between words.

When it comes to LDA and LDA-U, we expected LDA-U to perform better. The motivation is that when the messages are aggregated the issue with sparsity decreases. According to the results, there is not a large difference between these models either. Looking at the F1-scores in Table 4.10, it shows that LDA has a better score in general for the topics with the most template messages, while LDA-U could handle the topics with less template messages better.

The conclusion we can take here is that when choosing a topic model, we cannot only

consider the length of the documents, but also the structure of the documents.

5.10 Further work

Our current way of clustering messages into topics may have worked fairly well, but it is simple and we did not explore this area enough. More sophisticated clustering algorithms such as DBSCAN (Ester et al., 1996) and hierarchical clustering maybe could have found other clusters.

In this work, we manually labeled the topics based on the top words. Next step, could be to do this automatically by using WordNet (Miller, 1995). In WordNet you can search for word hierarchies to find the common word that connects the top words.

Neural networks would be another interesting application of the results. For example, topic model could be used to label the dataset and by extracting the best results from the topic models it could be used to solve classification problems.

Building upon this work, one can implement an online version of the algorithm. This can lead to being able to analyze text message traffic in real time to predict potential trends in the communication market.

Chapter 6
Conclusions

Topic models have been evaluated as a good method to initially explore an unknown corpus and its content. Depending on the purpose of analyzing a dataset, topic models can suffice if exploration is the main goal. For further applications, it can also serve as a foundation, such as classification. In our case, we have explored our given dataset and found 15 unique topics spread over 25 topic clusters, which can be seen in Table 4.4.

Concerning the prevalence of each topic, we conclude that the corpus percentages of Tables 4.1-4.3 are not exact, but still indicate how much the corpus contains of each topic.

Overall, there was no model that was significantly better than the others on Sinch's dataset, but based on topic coherence scores, F1-score and the human evaluation there is an indication of that LDA-U is better in performance.

Bibliography

Abdi, H. and Williams, L. J. (2010). Principal component analysis.

Blei, D., Y. Ng, A., and Jordan, M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Boyd-Graber, J., Hu, Y., and Mimno, D. (2017). Applications of topic models. *Foundations and Trends® in Information Retrieval*, 11:143–296.

Chang, J., L. Boyd-Graber, J., Gerrish, S., Wang, C., and M. Blei, D. (2009). Reading tea leaves: How humans interpret topic models. In *Neural Information Processing Systems*, volume 32, pages 288–296.

Chuang, J., Manning, C. D., and Heer, J. (2012). Termite: Visualization techniques for assessing textual topic models. In *Advanced Visual Interfaces*.

Darling, W. (2011). A theoretical and practical implementation tutorial on topic modeling and gibbs sampling.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press.

Honnibal, M. and Johnson, M. (2015). spacy. [Online; accessed 3-May-2019].

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

McKinney, W. (2010). Data structures for statistical computing in python. In van der Walt, S. and Millman, J., editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Oliphant, T. (2006). NumPy: A guide to NumPy. USA: Trelgol Publishing. [Online; accessed 3-May-2019].

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pleplé, Q. (2013). Perplexity to evaluate topic models.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Resnik, P. and Hardisty, E. (2009). Gibbs sampling for the uninitiated.

Röder, M., Both, A., and Hinneburg, A. (2015). Exploring the space of topic coherence measures. *WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, pages 399–408.

Sievert, C. and Shirley, K. (2014). Ldavis: A method for visualizing and interpreting topics. *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 63–70.

Weng, J., Lim, E.-P., Jiang, J., and qi, Z. (2010). Twitterrank: Finding topic-sensitive influential twitterers. pages 261–270.

Yadav, S. (2018). K-means clustering of 1 million headlines.

Yan, X., Guo, J., Lan, Y., and Cheng, X. (2013). A biterm topic model for short texts. In *WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web*, pages 1445–1456.

Appendices

EXAMENSARBETE Clustering Short Text Messages Using Unsupervised Machine Learning
STUDENTER Myky Tran, Michael Truong
HANDLEDARE Pierre Nugues (LTH), Lars Novak (Sinch), Jianhua Cao (Sinch)
EXAMINATOR Jacek Malec (LTH)

Vad tror AI att pratar alla om?

POPULÄRVETENSKAPLIG SAMMANFATTNING **Myky Tran, Michael Truong**

Idag skickas det ett oräkneligt antal SMS runt om i världen, men vad handlar dessa SMS om? På senare tid har man utvecklat modeller som kan analysera stora mängder text. Modellerna hjälper oss att hitta ämnen såsom "sport" och "forskning". Hur coolt hade det inte varit att ständigt vara uppdaterad om de senaste trenderna?

Modellerna som används för att hitta dessa trender kallas för *topic models*. Författarna har använt tre modeller som heter latent Dirichlet allocation (LDA), biterm topic model (BTM) samt en variant på LDA som kallas LDA-U. Dessa är statistiska modeller som grupperar ord baserat på hur ofta orden förekommer tillsammans. Varje ämne (gruppering) representeras av flera ord, som tillsammans uppfattas som ett ämne av oss människor. I bilderna ser vi två exempel som en av våra modeller gett oss, baserat på nyhetsrubriker från Australien. Kan du gissa vilka ämnen de syftar på? Tjuvläs inte under bilderna!

Inom industrin appliceras dessa modeller på långa texter för att exempelvis automatiskt kategorisera ett dokument eller en hemsida. För korta texter är det en större utmaning och något som man forskar intensivt på. Problemet med korta texter är att längden gör det svårare att hitta ett sammanhang.

Inom telekommunikationsindustrin, skickar företag massor av SMS varje dag för diverse ändamål, alldeles för många för att kunnas läsa igenom manuellt. Genom att applicera topic models kan man sammanfatta de SMS som skickas. Dessa sammanfattningar kan ge insikter både om vad SMS:en handlar om och hur mycket det pratas om varje ämne. Mäter man detta över längre perioder kan man hitta intressanta trender som företag kan agera utefter.

I examensarbetet har författarna lyckats hitta 15 ämnen från ett dataset av företags-SMS. De modeller som testades presterade ungefär lika bra, men den som var bäst överlag var LDA-U.

Svar: Bilderna handlar om handel och val.