University of Amsterdam

MSc Artificial Intelligence
Master Thesis

Finding Frequently Asked Questions

by
David Zomerdijk
10290745

July 5, 2018

Supervisor:
Dr. Miguel A. R. Gaona

Assessor:
Dr. Jelle Zuidema

Abstract

Large companies can receive thousands of questions and complaints each day. To improve the customer experience it is important to understand which questions are frequently posed such that problems can be solved and unclarity can be removed. Because the number of questions is so large it is not possible to read all of them. In this research we investigate how to automate the process of finding the most frequently asked questions (FAQ). We propose to tackle this problem by using a topic model to find the different types of questions after which we condition an LSTM based language model on the individual topics to generate a sentence that resembles the frequently asked question.

We explore several topic models to determine which model is most suitable to cluster questions. We try LDA , NVDM, GSM and Prodlda (David M. Blei, Andrew Y. Ng, and Michael I. Jordan, 2003; Miao, Yu, and Blunsom, 2016; Cao, 2015; A. Srivastava and Sutton, 2017). Because questions can be considered short documents we also propose an LSTM based encoder for Prodlda to improve its performance for short documents. We conclude that LDA is the most robust topic model because it achieves the highest topic coherence across almost all datasets. From the neural topic models Prodlda achieves the highest topic coherence. For documents shorter than 30 words our proposed model achieves the highest topic coherence. Hereby confirming our hypothesis that information about word order becomes increasingly important for shorter sentences.

To generate the most frequently asked questions we explore three ways to condition a neural language model: a mixture of experts approach, concatenating the topic vector to the input, and by concatenating the topic vector to the LSTM output. From the three approaches only the latter method was able to generate grammatically viable sentences that resemble the topic. We also find that this model was able to achieve lower perplexities than the standard language model by leveraging the global information about the document.

To determine whether the model is able to generate the FAQ we apply it to a proprietary dataset from the ABN AMRO bank that contains around 100k descriptions of call center conversations per month. The topic model was able to find topics and the language model was able to generate text that had a clear connection to the topic. However, it was not able to give a proper list of the FAQ. We believe that the dataset played an important role in the poor performance of the model. Additional research is required to determine whether our approach is able return a better list of FAQ with a dataset that only contains questions instead of descriptions.

Acknowledgements

Writing this thesis was not always easy, but luckily there were people I could fall back on. This small section is devoted to those people.

Firstly, Miguel, my supervisor, who really dedicated a lot of time to help me and made sure I understood the theory behind the models. Jelle, for assessing my thesis and helping me to find a supervisor.
Julia, who gave me the opportunity to come to ABN AMRO.
The people at ABN AMRO who made sure I kept working and made it more fun: Wido, for the morning talks, Seb for the countless hours of Fussball, Jurjen for the Jeu de Boules games and Jochem for the Nerf gun fights.
Thijs, my classmate, for being a listening ear after squash matches.
My parents on who I can count and made sure I could go to university and supported me in every decision.
And lastly Gillian, my girlfriend, who had to deal with me after days that were less successful.

Thank you.

List of Abbreviations

AI Artificial Intelligence

GSM Gaussian Softmax Model

LDA Latent Dirichlet Allocation

LSI Latent Semantic Indexing

NPMI Normalized Pointwise Mutual Informationl

NVDM Neural Variational Document Model

NVI neural variational inference

pLSI probabilistic Latent Semantic Indexing

VAE variational autoencoder

Contents

Chapter 1

Introduction

As a company understanding your customer is essential to provide them with the best experience possible. One way to improve your understanding is by knowing what questions and complaints they have. For a small bakery this is straightforward, the owner receives all the questions and complaints and therefore has a good overview of the customers needs and desires. As a consequence the baker can make a decision on what to change to improve the experience.

For a larger company this is more difficult. There is not one person that has the overview because they receive thousands of questions each day which are answered by a team of people. To still be able to make informed decisions on how to improve the customer experience, the company needs to find the set of most frequent questions and complaints.

For simplicity, we reduce the problem to finding the most frequently asked questions (FAQ). But how to tackle this problem? We can think of creating a set of labels that represent different types of questions and assigning each question one of the labels. The results is an overview of the frequency of every type of question. This will work well at the start, but quickly one finds that there are new types of questions which will cause a proliferation of labels. This will make it harder for employees to find the correct labels for the question and also probably cause overlapping labels.

For the above reason we look for ways to automate this process and create a system that finds the most frequently questions itself. Since similar questions are formulated in different ways, simply grouping them based on the exact text is not an option. Also building a rule based system to separate questions does not scale for the same reason as assigning them labels. For this reason a data driven approach seems to be necessary to build a scalable system that is able to find the FAQ.

In this thesis we will propose several methods that combine existing machine learning models to find the most frequently asked questions. We will explore state-of-the-art neural topic models to find separate questions within a dataset of call center conversation descriptions. Because topic models only represent the topics using a word distribution we also explore neural language models to generate text that represent the found topics. By combining topic models and language models we try to generate a set of sentences that represents the FAQ. Finally, we propose an adjustment for one of the topic models which improves its performance for short documents. The latter is useful for our data since questions are usually shorter than the average documents used with topic models.

To make our experiments reproducible all code for the models and experiments on public datasets have been made available on `https://github.com/DavidZomerdijk/Finding-Frequently-Asked-Questions`.

1.1 Collaboration with ABN AMRO

From the get go I have been excited to write my thesis at a company. Since I don't aspire a PhD, it seems to be the best way to prepare for future employment. But which company to choose? This turned out to be a challenge in itself. The first step is having coffee with lots of potential companies. I visited Philips Healthcare, Frosha, Deloitte, ABN AMRO, ING, and Textkernel. In the end ABN AMRO turned out to be the best company for me because they supplied me with the most exciting project and a lot of freedom on how to approach it. During my thesis I was part of the Data Innovation and Analytics department which is occupied with applied AI research as well as the more complex data-science projects. The bank provided the data, a workspace, a laptop and a group of data scientist who provided me with fun and mental support. Since the data contains sensitive and personal information the data was never allowed to leave the premises. However, this research and the content of this thesis has been checked by Wido van Heemstra, the head of data innovation and analytics, and is allowed to be read by anybody without any form of non disclosure agreement. In table 1.1 an overview of this information is depicted.

1.2 Problem Setting

ABN AMRO's call-center receives around 130 thousand calls each month. The call center operators have a free text field in which they describe the content of the call. Currently the business is interested in the question: *"what are the most frequently asked questions from our customers?"*. Since the call center operators can formulate the same question in many different ways, the questions can't simply be aggregated to find this. The business already tried looking at word frequencies but this didn't yield satisfying results. Hence, the problem for us to solve is: how to take the input from this open text field and find the most frequently asked questions/complaints/requests. A schematic drawing of the problem setting can be found in figure 1.1.

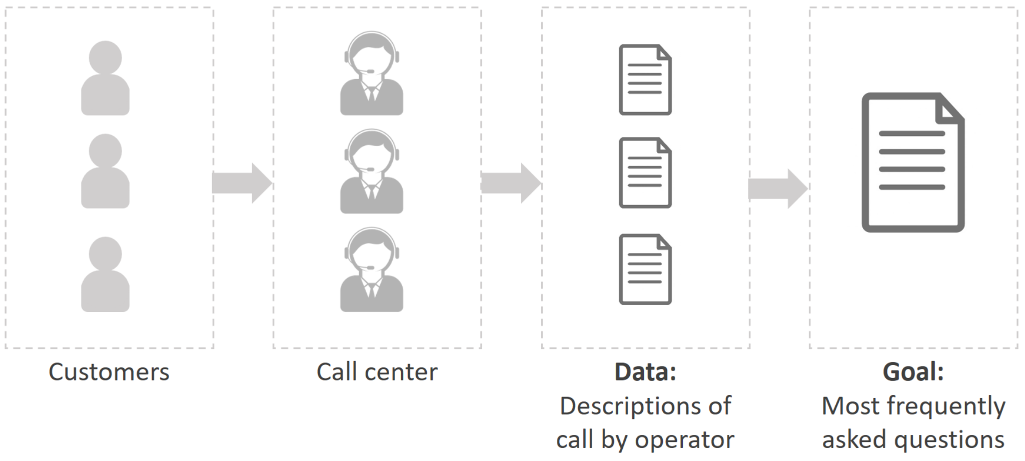

Customers
Call center
Data:
Descriptions of call by operator
Goal:
Most frequently asked questions

Figure 1.1: A schematic drawing of the problem setting.

At this point one might think: *"but what if the call operator just assigns every call to a type of question?"*. This is actually already being done to some extent, however, the groups are still considered to be too big to be useful. Currently every call is assigned to a `product` a `subject` and a `symptom`. On top of this the action required to fix he problem is also assigned. Another problem with the manual way of defining groups is that it leads to overlapping topics and misclassification by the operator.

Obviously this is a specific case in which the call center gives an extra dimension to the problem. If we disregard the call center operators it becomes a common problem that every company and organization faces: finding the most frequently asked questions from the set of questions they receive.

1.2.1 Data ABN AMRO

Now the general problem setting is clear, let's have a look at the data. The data is stored in a relational database and every row represents an interaction with the customer. In total 42 variables are logged of which one one important to us: the free text description by the call center employee. In table 1.2 one finds a few examples that represent the kind of descriptions given. As one can observe they range from simple 1 sentence summaries to longer descriptions with a lot of information about the specific case and personal information about the client. Most descriptions are in Dutch and the average text length is 180 characters and 24 words. In figure 1.2 the distribution of description length is depicted and we see that most descriptions are shorter than 24 words. The data-dump from the database we worked with contains 7 months of data consisting of 1,095,997 rows. After removing duplicates and empty descriptions fields, 938,412 rows were left for us to use.

1.3 Research Questions and Contributions

In this thesis we aim to develop a method that is able to give insight in the most frequently asked questions from a dataset, as described in the previous section. Although the data from ABN AMRO is rather specific

Type	Descriptions from call center
Avg monthly calls	134,000
Unique call operators in dataset	5790
Timespan data	Jan-2017 : Sep-2017
Avg description length	180 characters
Avg description length	24 words
Language	Dutch

Table 1.1: Description dataset

1	*limiet betaalpas met klant via ib aangepast.*
2	*Klant wilde limiet verhogen van 500 naar 1.000*
3	*Pas geblokkeerd van vader voor IB.*
4	*Mevrouw heeft via de rechterlijke weg een derde naam erbij. Ze wil graag al haar bankgegevens hierop aangepast hebben. geadviseerd met haar geldige en aangepaste paspoort langs te gaan op kantoor*
5	*Telefonie Bc nummer: <number>Bak: CM Wait Gesproken met: Mevr <name>Telefoonnummer: <number >Brief + bijlagen nogmaals sturen naar adres van vorige navraag SK <number>Datum brief: <date>Locatie:<address>Reden: Mevr belde net dat broer had gebeld en dat hij een Nederlandse brief heeft ontvangen. Mevr uitgelegd dat hier in ons systeem wel degelijk een Engelse brief staat. Mevr vond dit ook heel raar, maar vroeg of wij het nog eenmaal dan willen versturen naar het adre sin <country>. To the estate of the late Mrs. <name><address><country>*

Table 1.2: Example Messages

because it is generated by the call center operators instead of directly by the customers directly, the research will be mainly focused on the general case where one would have a dataset of questions directly posed by the customer. By doing so we hope this research can benefit a larger group, since practically all companies and organizations receive emails with questions.

This research is split into two parts. The first part will explore all possible methods to develop such a method and determine which is most interesting to focus the rest of our research on. The second part will be focused on implementing and further exploring this method.

Underneath you find the research questions that will be addressed throughout this thesis. For obvious reasons the second part of the questions have been formulated after the first research question was answered.

Part 1

RQ 1 How to find the most frequently asked questions from a large set of questions?

Part 2

RQ 2 Is a topic model a suitable choice to distinguish different types of questions?

RQ 3 What topic model is the most suitable choice to distinguish different types of questions?

RQ 4 Is a language model conditioned on the latent space of a topic model able to generate the most frequently asked questions?

After having answered all four research questions in the thesis we can distinguish three contributions that this thesis offers to the field:

1. We propose an extension on the neural topic model Prodlda which makes it perform better on short documents.
2. Applying existing models to a new type of data results in more insight into the strengths and weaknesses of the model. Especially because we apply recently proposed methods, such as Prodlda and topic conditioned LSTM cells, this is useful for others who consider using these models.

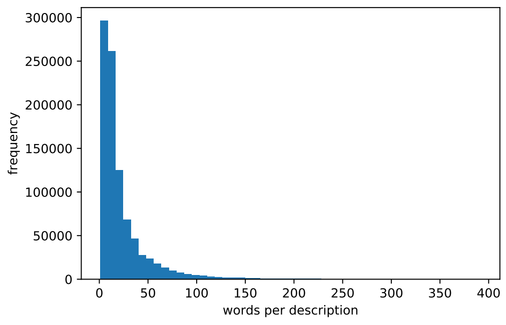


Figure 1.2: Distribution of description length.

3. We are the first that address the problem of finding the FAQ using machine learning. This can pave the way for future applied machine learning research.

Lastly, our hypothesis. Without any context it is difficult to establish an hypothesis. For this reason we take the first research question to provide ourselves with an overview of the options. After considering all options we arrive at the following hypothesis: "A combination of a topic model with a language model is an effective option to find the most frequently asked questions". All mentioned research questions serve the purpose to either accept or reject this hypothesis.

1.4 Overview Potential Methods

To answer the first research question, *"How to find the most frequently asked questions from a large corpus of questions?"*, we first need to know what our options are. In this section we will consider all possible options in an overview. Every option will be briefly assessed after which we will have a closer look at the most promising methods in the next section.

In table 1.3 an overview of all possibilities are depicted with their advantages and disadvantages. From this list of methods `Supervised Classification`, `Summarizing Clusters of data` and `Topic Model + language model` seem to be the most promising methods based on their feasibility, novelty and effectiveness. Feasibility is important because we want to end up with a working model, novelty is important to make it interesting from a research perspective, and effectiveness because the model has to solve the problem as good as possible.

In the next chapter we will have closer look to see which method is most interesting to explore in this thesis.

Method	Advantages	Disadvantages	Description
Supervised classification	1) many proven models 2) predictable outcome	1) requires labeled data 2) only fixed targets	Pose the problem as a classification problem where we have a fixed group of questions.
Summarization	1) finds abstract texts that captures question	1) requires labeled data 2) no previous work	view all documents as one big document and summarize it.
Information Retrieval Methods	1) no annotated data required 2) useful for exploration	1) only useful for exploration	Use Information Retrieval to find the number of relating documents
Cluster paragraph Embeddings	1) possibly better than clustering BOW 2) no annotated data required	1) no theoretic base	Create semantic clusters by clustering paragraph embeddings.
Topic Model	1) proven to be good at finding semantic topics 2) no annotated data required	1) only generates most likely words	Use most likely words per topic to determine what kind of questions exist.
Language Model	1) can generate sentences that describe dataset 2) no labeled dataset required	1) similar effect as just reading the questions	train a language model on all questions and generate sentences that represent the data.
Topic Model + Language Model	1) No labeled dataset required 2) generate most likely sentences given the topic	1) little publications in this domain	train a language model conditioned on topic model. This model can then be used to generate the most likely sentence given the topic.

Table 1.3: Options to find most frequently used questions.

1.5 A closer look into the most promising methods

In the following subsections we will have a closer look into the three methods that seem most promising in the previous section.

1.5.1 Supervised Classification

Within Machine Learning most research is focused on supervised machine learning. A quick, non scientific, search on Google Scholar confirms this. The search term "supervised machine learning" yields 905k publications against 303k for "unsupervised machine learning" (30th of April 2018). This doesn't come as a surprise since almost all AI applications are based on supervised machine learning, think of image classification, translation models and predictive models.

This problem can be easily posed as a classification problem by creating a dataset where each type of question is assigned to a class. By posing it in this way we create a situation where we can use a myriad of proven models to solve the problem. The easiest way would be to create a bag of n-grams of the documents and fit an arbitrary classifier to the data (e.g. a Support Vector Machine, Random Forrest or a feed forward Neural Network). Joulin et Al. (2016) show how the results of such a model are often on par with that of more complex deep learning models with shorter training times and even reach 98% accuracy on the DBPedia dataset.

However, for short documents it seems that deep learning methods outperform these methods and are an active field of research. Recursive neural networks, recurrent networks, convolution networks and combinations of these methods are researched to improve text classification (X. Zhang, Zhao, and LeCun, 2015).

The reliability of such models and the active research field make this a promising method to focus this thesis on. However, this method requires high quality annotated data. It takes a lot of time and resources to create such an annotated dataset which makes it a liability in the success of this thesis. On top of this, maybe the best reason not to choose this approach is because it doesn't solve our problem. The call center employees are already labeling the data, which apparently isn't enough. For us to solve the problem we need a system that can find the classes itself.

1.5.2 Summarization

The second possibility we explore is automatic summarization. In this situation we perceive all questions as one big document which we will then try to summarize automatically. By doing so we try to get a short summary of all received questions and complaints that allows the business to get insight into the interactions with the customer.

The advantage of this method is that it is the most intuitive one. Just like children are learned in school how to summarize documents, just learn a system to summarize the content of our stream of questions. However, it's not as easy as it seems.

Automatic summarization can be divided in extractive and abstractive methods. The extractive methods try to extract the words and phrases that it deems to have the most important information. Abstractive methods are more similar to how a human would summarize a text, namely creating a shorter text that covers the semantic content with words that don't have to necessarily overlap.

Extractive summarization methods have been the most common in the past (Nallapati, Zhai, and Zhou, 2017a). However, most recent methods have been focused around abstractive methods (Paulus, Xiong, and Socher, 2017). The approach for both methods is currently mainly focused around deep learning. However, still a fair amount of research is focused on graph based methods (Erkan and Radev, 2004) (L. Zhang et al., 2016).

Although most research is focused around deep learning models, such methods need a training set with summaries of documents. To our knowledge there is no public summary dataset for banking data which makes deep learning based summarization models unlikely to be successful. Graph based methods could still be an option because they don't require a training set for it to extract meaningful insights.

Summarization also has a few other disadvantages. Firstly, all quantitative information is lost: the summary will not mention how often a question comes back which makes it impossible to prioritize the questions. Secondly, supervised summarization will have a bias towards the training set, and just like with human summarization, different persons find different aspects of a document important.

Concluding, graph-based summarization methods seem to be the best approach because we don't possess a proper dataset. The downside of this approach is that the author has no experience in graph based methods which makes the outcome of a thesis in this area uncertain.

1.5.3 Language Model conditioned on a Topic Model

In the third option we explore the combination of topic models and language models. In this scenario we have a topic model that finds the topics within the documents, but instead of using the most likely words given the topic to represent the topics, we have a language model that generates a piece of text given each topic.

Topic models have been around for quite some time. The most common method is Latent Dirichlet Allocation (David M. Blei, Andrew Y. Ng, and Michael I. Jordan, 2003). This is a generative model that fits a distribution of topics to a document. The method is popular because the topics are expressed as distribution over words which makes them interpretable. It is also popular because it doesn't require any annotated data because it's trained in an unsupervised manner.

One of the main challenges in applying topic models is the computational cost required to find the posterior distribution. This is why a lot of research is done in approximate inference methods to find this posterior, such as variational inference (David M Blei, Kucukelbir, and McAuliffe, 2017). Kingma & Welling (2013) introduced neural variational inference by proposing the variational autoencoder (VAE) which uses a neural network to approximate this posterior. Soon after the introduction of the VAE, Miao et Al. (2016) identify the VAE as a potential way to make topic modeling easier and more scalable. It turned out to be harder than expected due to various instabilities, however, several neural topic models currently exist that outperform LDA. One of them is Prodlda which is a method that uses neural variational inference to train LDA (A. Srivastava and Sutton, 2017). The method is not only more scalable due to it's ability to be trained in parallel on GPU's, it is also able to systematically find more coherent topics. The disadvantage of topic models is that topics are often hard to interpret. Hence, using language models to generate text that represent the topic is potentially a way to overcome this and create actual topic descriptions.

In the last few years recurrent neural network (RNN) based models have become the standard in language modeling due to its ability to effectively learn long and short term dependencies (Hochreiter and Schmidhuber, 1997). Recently, there also have been several successful attempts where a language model is conditioned on a topic model (Adji B. Dieng and Paisley, 2016; Wang et al., 2017). However, all the research has been focused around topic modeling datasets, and no examples exist where it used to find the FAQ. On top of this, there are no public implementations available of conditioned LSTM's which means we have to implement it ourselves based on the equations from the papers.

Concluding, combining topic models and language models seems to be a promising approach solve our task of finding the FAQ. It also provides a challenge because it requires to understand topic models, neural variational inference and RNN based language models. In addition, because all proposed methods are very recent there is enough left to be explored.

For the above mentioned reasons this thesis will further focus on combining topic models with language models. Especially due to the limited time frame this methods seems to have the most potential to deliver value for the data provider ABN AMRO.

In later chapters we will go in more detail how the topic and language models work. Also, a more extensive overview of previous work can be found in the section *related work*.

1.6 Related Work

In this section we give an overview of research that addresses our problem as well as research that uses similar techniques as we do. We will discuss the more specific relevant work, such as related topic and language models, in the next chapters.

Even though finding the most frequent questions seems relevant to most companies, organizations and governments, there is little research addressing this. One of the few papers addressing this problem is about clustering community complaints in Jakarta (Dhino and Hardono, n.d.). To achieve this they use self organizing maps and limit themselves to 6 topics per department to keep an overview. The number of complaints per cluster are then used to prioritize the problems. Most other research is focused on classifying complaints. For example, another paper from Indonesia uses k-Nearest Neighbors to classify which complaints should be forwarded to which department (Warsito and Sugiono, 2016).

Our thesis is the first that use topic models to cluster questions and complaints, and use a language model that generate text that resemble the clustered complaints. Nevertheless, topic and language models have been researched extensively, this will be discussed in more detail in the next chapters. Also the combination has been recently proposed by several research groups.

Tian et Al. (2016) from Microsoft Research are the first to introduce a model that use an RNN to, as they call it, "Let the topics speak for themselves". Their model is called Sentence Level Recurrent Topic Model (SLRTM) and assumes the generation of each word depends on both the topic of the sentence and the preceding words in the sentence. Another model that uses an RNN to generate text to represent the topics is Sentence Generating Topic Model by Nallapati et Al. (2017). This is a topic model that assumes that every sentence has the same topic. The generative process therefore samples a topic for every sentence, and then, sample a sentence given the topic using an RNN. This model is trained end to end and learns the topics using a variational autoencoder (VAE).

Another research by Microsoft's researchers is TopicRNN, this model also use RNN's and are able to generate text given the topic. However, this research is more focused on reducing the perplexity of a language model using topics (Adji B. Dieng and Paisley, 2016). Lastly, the Topic Compositional Neural Language Model is also a language model that leverage a topic model. They use a Mixture of Experts approach to create a better language model where each language model is specialized for a topic. This research also propose a model that self-tunes the number of topics in the topic model by optimizing the number of topics for perplexity (Wang et al., 2017).

Our research complements this research by providing an overview of the techniques, create a model that works better on short documents and applying it for a specific goal.

Chapter 2

Technical Background

In this chapter we will look into the technical background of the methods we use in this thesis. We start with generative models, which are necessary to understand current topic models. Then we will go into neural networks, which are the core of our language model and some of the topic models. And lastly we will discuss the variational autoencoder (VAE) and how it can be used to train a generative model.

2.1 Generative Models

Within Machine learning there are two main approaches, A generative and a discriminative approach. The discriminative approach is most common approach and methods like SVM, Logistic Regression and Multi Layer Perceptrons fall under this approach. They are based on the premise of conditional probability, which is the probability for the target Y, given observation X: $P(Y|X = x)$. Although the mentioned methods might not explicitly model this probability, this is how people use the models intuitively. The generative models on the other hand try to find the joint probability of the target variable Y and observed variable X, $P(X, Y)$ (Andrew Y Ng and Michael I Jordan, 2002). Examples of generative models are Mixture Models, LDA and Naive Bayes. For this thesis we are mainly interested in Generative models for which there are no targets and the model describes a process where the data X is generated from a latent variable Z. This latent variable Z is often assumed to also come from a specific prior distribution. Since these models can get mathematically complex, these models are often expressed using graphical models that offer insight into the conditional dependencies of the model. The advantage of these models is that they explicitly model $P(X|Z)$ which allows us to know the probability of x given the latent variable z. For the topic model this is very useful since it allows us to get the most probable words x, given the topic z.

Inference in such a generative model amounts to conditioning the latent variable on the data to compute the posterior $p(z|x)$ (David M Blei, Kucukelbir, and McAuliffe, 2017). To then approximate the posterior we try to find q*(**z**) that is as close to the posterior as possible.

$$q^*(\mathbf{z}) = argmin(KL(q(z)||p(z|x)) \tag{2.1}$$

To do so we try to minimize the KL divergence between q(**z**) and the posterior $p(z|x)$. However, this objective function is not computable since $p(z|x)$ depends on finding the evidence $p(x)$ (Bayes rule). The reason $p(x)$ is intractable is because attaining p(x) requires marginalizing over all options of z ($p(x) = \sum_z p(x|z)$). Since z is continuous the summation is intractable. For this reason we usually optimize an alternative objective equivalent to the KL with an added constant, such as the evidence lower bound (ELBO).

$$ELBO(q) = \mathbb{E}[logp(z, x)] - \mathbb{E}[logq(z)] \tag{2.2}$$

The former can be optimized in many ways, the most popular methods are variational methods, especially mean field methods, and Markov chain Monte Carlo, especially methods based on collapsed Gibbs sampling (A. Srivastava and Sutton, 2017). Lately Stochastic Gradient Variational Bayes (SGVB) which uses a variational autoencoder to approximate the posterior has gained popularity (Diederik P. Kingma and Welling, 2013). We will use the latter method in most of our topic models which is why we have dedicated a specific section to this method.

2.2 Neural Networks

In this study we use neural networks (NN) in both our topic model and language models. This section therefore serves to provide some background on these models.

The goal of a NN is to map an input X to an output Y using a combination of linear transformations and non-linear activations. The most basic neural network, a multi layer perceptron, can therefore be generalized as:

$$\hat{y} = f_L(..., f_2(f_1(x, \theta_1), \theta_2)) \tag{2.3}$$

Where $\hat{y}$ is the approximation of the target by the neural network. f is a non-linear activations function such as a sigmoid, tanh or a rectified linear unit and θ_l are trainable weights.

The weights are optimized in such a way that the loss over a training set is minimized. The loss is a function which contains the difference between the target and the output of the neural network. By back-propagating the error and changing the weights in the direction of the error we can optimize the weights for the training set. This update of the weights is known as back-propagation and is formulated as:

$$\theta^t = \theta^{(t-1)} - \eta_t \nabla_\theta \mathcal{L} \tag{2.4}$$

Where θ are the trainable weights η the learning rate which governs the rate by which the weights are adjusted and $\mathcal{L}$ the loss.

There are many choices when using a neural networks such as number of layers, number of hidden neurons, how to initialize the weights, which optimizer to choose and how to regularize the model. Below we will discuss these in more detail.

Weight Initialization
The trainable weights have to be initialized before the network is used. Bad initialization can result in poor performance. When the weights are too small the network will have difficulty learning the data because the update signal is too small. When the weights are too big the update signal becomes too large and the model might overshoot the optimum. It is also important that the weights are asymmetric to prevent them from learning the same.

The method we will use to initialize our models is Xavier initialization which has proven to work well (Glorot and Bengio, 2010). This form of initialization samples the weights randomly from a normal distribution given by:

$$\theta = \mathcal{N}(0, \frac{2}{n_i + n_{i+1}}) \tag{2.5}$$

where n_i are the number of nodes in the current layer and n_{i+1} are the number of nodes in the next layers.

Activation Function
There is a myriad of activation functions available: sigmoid, tanh, rectified linear unit (ReLU), leaky ReLU, Swish and many more (Krizhevsky, Sutskever, and G. E. Hinton, 2012). There can be large differences in performance between the activation functions which makes it important to choose the right one. In our thesis we will use the ReLU activation function because it is known to consistently score well in most situations, as well as that is a computationally efficient since both the output and its derivative are straightforward:

$$ReLU(x) = \begin{cases} x & \text{if x > 0} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

Optimizer
There is an entire field of research focused on how to optimally change the weights for a neural network. Traditionally people used gradient descent where the errors were propagated based on the entire dataset. Later people found that by updating the weights based on a single data point was a lot more efficient. This is known as stochastic gradient descent. Since GPU's can do the matrix multiplication in parallel, people started to use mini-batch gradient descent to make the direction of the gradient more stable without having to trade-off for time.

Besides the number of data-points there have been other additions to model gradient descent which lead to a lot of available optimizers. We will use *Adam* to optimize our model because it combines both an adaptive learning rate and momentum into one optimizer (Diederik P Kingma and Ba, 2014). Adaptive learning rate speeds up the learning by starting with a large learning rate that allows to quickly get close to the minimum and slowly decrease the learning rate to be able to get as close to the minimum as possible without overshooting. Momentum makes sure that the model doesn't get stuck in a local optimum by keeping keeping some of the direction of the former gradients.

Regularization
When creating a model the goal is not only to perform well on the dataset, but also on new data points. To avoid that the model learns the data set by heart, we try to make this difficult for the model by using regularization. Two commonly used techniques to do this are L2-regularization and dropout. L2-regularization adds a penalty to the loss for the weights θ. This will discourage the model from learning functions that are too complex. Dropout regularizes by setting the output of a node to zero with a certain probability during training. By doing so the network is forced to not depend on a single node for its decision (N. Srivastava et al., 2014).

2.3 Variational Autoencoder

Kingma and Welling (2013) propose a new way to approximate the posterior of a generative model using a neural network, which they coin the *variational auto-encoder* (VAE). The VAE uses a pair of neural networks. One to generate the latent variables from the data and another to generate the data from the latent variables, this process is depicted in figure 2.1.

A standard auto-encoder has an MLP encoder that encodes the input to a lower dimension. A MLP decoder then takes this lower dimensional representation and tries to reconstruct this to the original input. The network is trained by propagating the error, which is the difference between the input and the output of the decoder, through both the decoder and encoder. The disadvantage of this method is that it has the tendency to overfit on the training data and that the space of the lower dimensional representation has no meaning.

The VAE is different from the standard auto-encoder in two ways. Firstly, the latent vector z is sampled from a Normal distribution $z \sim \mathcal{N}(\mu, \sigma^2)$ where μ and σ^2 are learned by the encoder. Secondly, the loss is different. The loss is based on minimizing the ELBO, like discussed in the section on generative models, and consists of a reconstruction part $\mathcal{L}_{recon,n}$, and a regularization term $\mathcal{L}_{reg,n}$. The reconstruction term penalizes the output if it doesn't match the input. The regularization term penalizes the model when the latent space doesn't follow a Normal distribution. Combined the loss becomes:

$$\mathcal{L} = -\frac{1}{N}\sum_{n}^{N}(\mathcal{L}_{recon,n} + \mathcal{L}_{reg,n}) \tag{2.7}$$

where the reconstruction loss and regularization loss are:

$$\begin{aligned} \mathcal{L}_{recon,n} &= \mathbb{E}_{z\sim q(Z|X=x_n)}[logp(X = x_n|z)] \\ \mathcal{L}_{reg,n} &= KL(q(z|X)||p(x)) \end{aligned} \tag{2.8}$$

One of the challenges in the model is the sampling step. If implemented naively, the computational graph will not function because it cannot pass a stochastic derivative. Kingma and Welling (2014) propose the Reparametrization Trick to deal with this problem:

$$\mathcal{N}(\mu, \sigma^2) = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0, 1) \tag{2.9}$$

The trick makes the computational graph differentiable by making the error only dependable on μ and σ.

Another benefit of the VAE is that the entire space of the latent vector becomes meaningful because it is sampled rather than merely passed through. This allows us to sample from any point in the latent space. Bowman et Al. (2015) use a recurrent VAE that encodes and decodes sentences using a VAE. They show how decoded sentences slowly change when moving through the latent space. This would not have been possible with a regular auto-encoder where points in the vicinity of one another yield completely unrelated sentences.

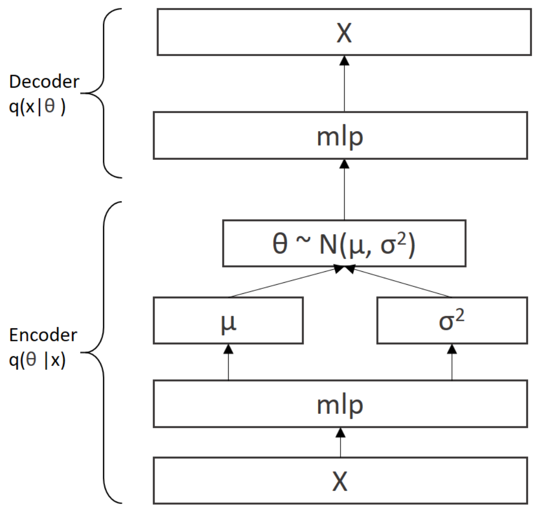


Figure 2.1: Variational autoencoder where θ represents the latent space z.

Chapter 3

Topic Model

In this chapter we focus on answering the second and third research question. This means we start by determining *"Is as topic model a suitable choice to distinguish different types of questions?"*, after which we will look at *"What topic model is the most suitable choice to distinguish different types of questions?"*. To answer the first question we will have a closer look at what a topic model is, how it works and how it has been used in the past. To answer the second question we will first have a more detailed look at several topic models to see how they work, after which we will apply them on several datasets to see how they perform.

3.1 What is a Topic Model?

Let's start by defining what a topic model is. If one would ask the French philosopher Jean-Paul Sartre he would say that the essence of something is embedded in what it does (Sartre., 1946). This makes sense because there are several topic modeling approaches and only their application is the common denominator. Therefore this section will look at how one can apply the topic model rather than seeing how it works. later in this chapter we will look extensively in how the individual models do what they do.

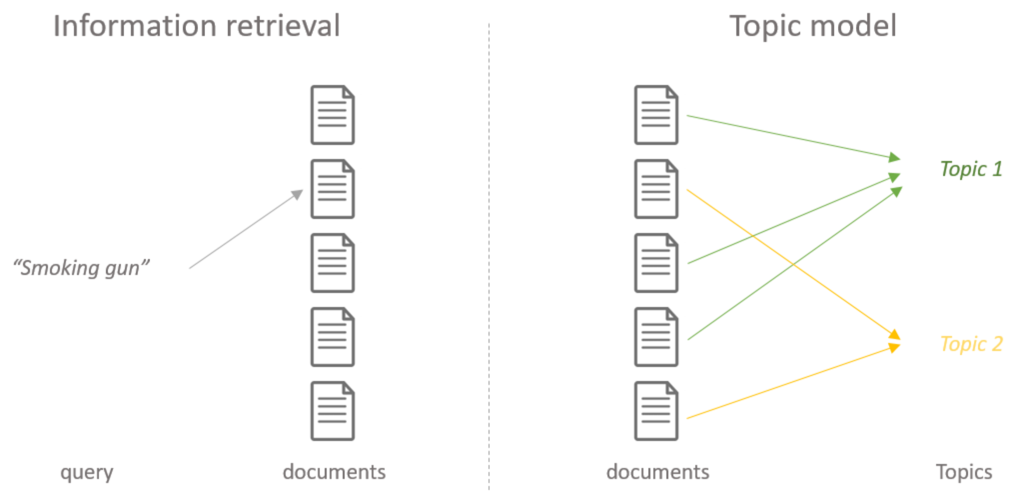


Figure 3.1: Simplified schematic drawing of information retrieval versus topic model. (it's simplified in two ways. 1) most topic models are mixtures of topics rather than assigned to a single topic. 2) Most topic models are generative which means that the assumptions require the arrows to be reversed)

Firstly, topic models enable users to interact with large text data sets (David M Blei and J. D. Lafferty, 2009). Working with large text data sets is difficult because as a person it is often impossible to read everything. Journalist are often confronted with exactly this problem when a large set of documents is leaked or made public. Probably the first approach the journalist would take is looking for answers to specific questions. This can be seen as looking for the needle in a haystack. The easiest approach is to look whether exact words or phrases are present in the corpus. However, this approach has many disadvantages. It could be that specific words occur very often in the text and it's only relevant in a few cases. Or it can be that the actual content you are looking for is formulated in a slightly different way. The field of Information retrieval has been focused on solving this by ranking the results and trying to find semantic matches rather than exact ones. This is depicted on the left side of figure 3.1 (Baeza-Yates, Ribeiro-Neto, et al., 1999).

But what if you are not looking for something specific but are interested in understanding the document set. So instead of looking for that needle in the haystack, you want to know of what straws the haystack consist. For example: if one wants to know what topics are being discussed on twitter and by what distribution they occur. At first, one might consider an approach where one builds an extensive ontology for the corpus. This implies that we need to define what buckets exist and when a document should be assigned to this bucket. Then one would need a team of people that use this ontology to assign every document to a specific bucket. One could even think of manually labeling 20% of the documents and use supervised methods to automate the remaining part of the process. Even though this sounds crazy to the average AI specialist this is exactly how librarians and archivists work (except for the automation part). This approach is both hard and costly. First one needs to define specific topics, and one often finds that a book doesn't fit only one but many topics. For example, if we would have a book about how birds are studied to improve airplanes, how should it be classified? As a book about birds? As a book about planes? This goes to show that even this costly, labor intensive approach doesn't yield perfect results. Secondly, the price associated with this approach makes the threshold to actually do it high, which leave many text data sets left unstructured. Luckily, we were not the first to realize this and this is exactly what Topic Models are for. They are used to assign topics to documents as depicted on the right hand side of figure 3.1.

So now we have determined that topic models assign topics to documents. But what is a topic? If a person would make these buckets it would name the bucket using an abstract concept we came up with like birds or planes. Unfortunately, topic models are not able to assign these abstract concepts to a topic. However, most topic models are probabilistic and are able to give the most likely words given the topic $p(w|t)$. So although the topic model is not able to name the topic, it tells one which topic(s) are associated to the document as well as words that are associated to the topics. In figure 3.2 one finds an example from the paper "Latent Dirichlet Allocation" where four topics are depicted. By attaching a language model to the topic model we hope to remove this last manual step where one needs to assign abstract concepts to the latent topics.

"Arts"	**"Budgets"**	**"Children"**	**"Education"**
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

Figure 3.2: Four topics from the LDA topic model on the AP news dataset. The words in quotation marks are the abstract topics assigned to the latent topics by the author. The rest of the columns represent the most likely words given the latent topics. (David M. Blei, Andrew Y. Ng, and Michael I. Jordan, 2003)

The only question remaining is: how does the topic model finds its topics? Since topic models don't have one specific way of working we can't get too specific and details will be discussed in later sections. But we can say that it's fundamentally different from the way that we, humans, determine topics. Where humans try to make a logical ordering which is easily interpretable for others, topic models have no knowledge of such concepts. The topic model consists of a set of parameters which are fitted in such a way that they represent the dataset as accurately as possible. Since there is no logical boundary on the topics this means that some topics found by the topic model don't make sense to the human reader. For example, if one doesn't remove the numbers from the 20Newsgroup dataset, one of the topic will be solely representing numbers, which is not a topic that humans would distinguish.

So what does this mean for our second research question: "Is a topic model a suitable choice to distinguish different types of questions?". On the one hand we have determined that topic models can efficiently structure large sets of text documents. On the other hand we have determined that topic models structure the data in different way than humans do which can lead to uninterpretable topics. This means that a topic model is a suitable choice to distinguish questions as long as it is not essential that every single topic represents a valid type of question. In our case the output is used to find the most frequently asked questions so as long as most topics are interpretable this isn't a serious issue.

3.2 Historic context

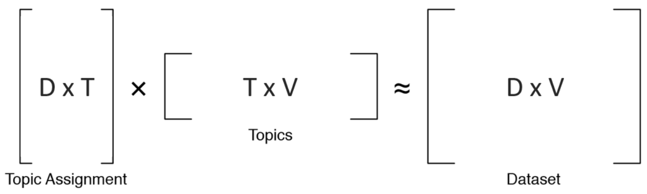


Figure 3.3: A depiction of how LSI works, where D represents the number of documents, T the number of topics and V the number of words in the vocabulary. The documents are represented as a bag of words.

As earlier mentioned, topic models find their similarity in their functionality rather than their inner workings. In this section we will show how the technique has developed over the years.

In Deerwester et Al. (1990) introduced Latent Semantic Indexing LSI which is perceived as the first topic model. This model, as depicted in figure 3.3, is a a linear algebra approach where a co-occurrence matrix of documents and words is decomposed into a document-topic matrix and a topic-word matrix using singular value decomposition. This approach has been mainly abandoned and replaced by probabilistic methods. However, it should be noted that there has been a small resurgence of linear algebra based topic models recently (Arora et al., 2013).

Nine years later Hoffmann was the first to propose a probabilistic alternative to LSI called Probabilistic LSI (pLSI), which is depicted in figure 3.4 (Hofmann, 1999). pLSI models each document d in the corpus as a mixture of topics. Then for every word in the document a topic is sampled after which a word is sampled from that topic. Also in this model a topic is represented by a multinomial distribution over words. This model didn't necessarily provide any advantages over LSI, it did however lay the foundations for Latent Dirichlet Allocation (LDA). Although pLSI has a proper statistical foundation, it uses the dummy variable d that represents the index of the the document. This means that the model is able to give a topic distribution for a document in the train set, however, it is not able to do inference on an unseen document. On top of this the model has to keep separate parameters for every document in the train set which means that the number of parameters grow linearly with the size of the document set. These two shortcomings were addressed and resolved by Blei et Al. (2003) when they introduced Latent Dirichlet Allocation.

LDA is probably the most commonly used topic model and is the basis of most other topic models. It solved the shortcomings of pLSI by treating the topic mixture weights as a hidden random variable. This decouples the topic mixtures from the individual documents which makes the number of parameters independent from the number of documents in the corpus. Also, now the topic distributions are not coupled to the individual documents we are able to use the model for inference on new documents. In figure 3.5 the plate notation of LDA is depicted which will be further explained in the model section.

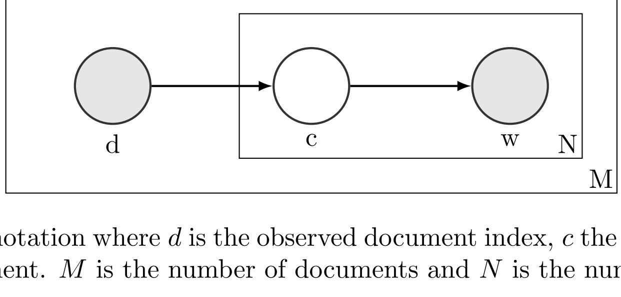


Figure 3.4: pLSI in plate notation where d is the observed document index, c the latent topic representation, and w the words in the document. M is the number of documents and N is the number of words in the document.

One of the main challenges in applying language models is the computational cost to find the posterior distribution. This is why most effort has put in to find the posterior more efficiently. Numerous methods have been proposed: Bayesian inference (David M. Blei, Andrew Y. Ng, and Michael I. Jordan, 2003) , expectation propagation (Minka and J. Lafferty, 2002), collapsed Gibbs sampling (Griffiths and Steyvers, 2004) and extensions on them (Teh, Newman, and Welling, 2007).

The latest innovation in topic models was spawned by the introduction of neural variational inference which approximates the posterior using a Variational Autoencoder. The first model to use this on text was the Neural Variational Document Model (NVDM) which uses a Gaussian to approximate the topic-document distribution and average over the topic-word distribution to find the most likely words given the topics (Miao, Yu, and Blunsom, 2016). Prodlda improves upon this model by imposing a Dirichlet prior over the topic-document distribution (A. Srivastava and Sutton, 2017). Although it seems like a small change, Wallach (2009) shows how important the Dirichlet prior is in creating interpretable topics. Also, Prodlda doesn't impose a multinomial over the topic-word distribution like LDA, this turns out to also result in more interpretable topics. Another neural topic model that has been proposed recently is the Gaussian Softmax Model by miao (2017), which passes the Gaussian latent vector through a softmax to parameterize the multinomial document-topic distribution.

Judging by this historical overview, topic models have not fundamentally changed since pLSI and improvements have been merely incremental since LDA. Nevertheless, due to neural variational inference it has become much easier to explore new types of topic models because it doesn't require complex derivations when working with new assumption. Hopefully this ease of exploration will lead to topic models that better mimic human topic selection in the future.

3.3 Models

In this section we will elaborate on the topic models discussed throughout this thesis.

3.3.1 LDA

Even though LDA has been around for many years it is still the most commonly used topic model. Throughout this section we will explore how it works as well as its advantages and disadvantages.

LDA is a generative model. This means that it assumes the documents in your dataset are generated from a process based on unobserved variables. In the case of LDA it assumes that every document has a distribution over topics, where topics are again a distribution over words. For every document a sample is taken from a Dirichlet distribution which yields a distribution over topics. Then for every word in the document a topic is sampled from this distribution after which a word is sampled from the distribution. Which can also be described as:

For every document M:

1. Choose N $\sim$ Poisson(ξ)
2. Choose $\theta \sim \text{Dir}(\alpha)$
3. For all N words in the document:
   (a) choose a topic $z_n \sim$ Multiniomial(θ_i)
   (b) Choose a word w_n conditioned from $P(w_n|z_n, \beta)$, where β is a multinomial distribution over words given the topic

The Dirichlet distribution is a convenient choice as a prior because it is the conjugate prior for the multinomial distribution which is used for the word distribution.

The key challenge with generative models is to find the posterior distribution of the hidden variables given the document:

$$p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)} \tag{3.1}$$

Although this distribution is intractable to compute there are several methods to obtain an approximate of the distribution. In the original paper Blei et al. use convexity based variational inference to obtain a lower bound on the log likelihood. A disadvantage of this method is that it is computationally expensive. Because of this

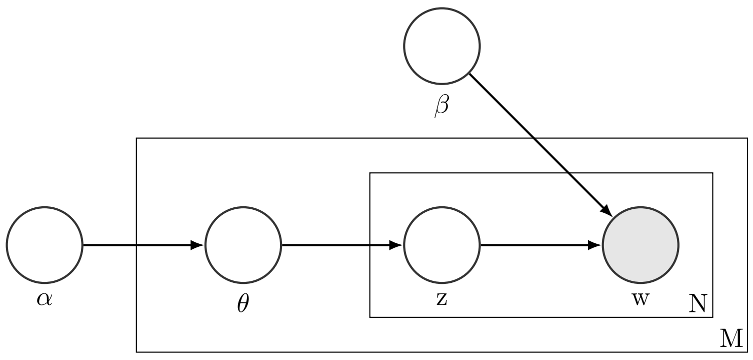


Figure 3.5: LDA. w is the word, z is the latent topic, θ is the topic distribution, α is a parameter of the dirichlet distribution that governs θ and β is the distribution over word per topic. N are the number of words in the document and M are the number of documents.

multiple other methods have been proposed such as Expectation Propagation (Minka and J. Lafferty, 2002) and Gibbs sampling (Griffiths and Steyvers, 2004; Porteous et al., 2008). Especially the latter is significantly faster and therefore widely used.

Next to the variational parameters, α and β are also parameters that can be tuned. These are the parameters that govern the shape of the distributions. Blei et al. (2003) propose using expectation maximization where the variational parameters and (α, β) are fitted alternately. In most cases however people do not fit these parameters even though it can yield significantly better results (Wallach, Mimno, and McCallum, 2009).

Despite of its success, LDA also has shortcomings. As discussed, finding the posterior is computationally challenging. Especially for larger datasets this can pose a problem. Another shortcoming is caused by how the documents are represented. The documents are represented as a bag of words which means that there is no knowledge on the word order of the document. This means it cannot distinguish between *"Fake cops robbed a bank"* and *"Cops robbed a fake bank"*. Same words, different meaning. Another disadvantage of the model that it is negatively effected by stopwords. For this reason it's common to use *tf-idf* to remove stopwords.

We will use LDA as a baseline for the the topic models.

3.3.2 NVDM

Neural Variational Document Model NVDM introduces a neural variational framework for text-based generative models. The basic concept of the model is to use a variational autoencoder (VAE) to learn the posterior probability of the topic model (Miao, Yu, and Blunsom, 2016). In this case the latent layer represents the topic distribution of the document. This approach has two advantages. Firstly, the neural network is able to learn complex non-linear relations between words in the documents. Secondly, this method can be fully parallelized and trained on GPUs making it more scalable than other methods that find the posterior.

In Figure 3.6 NVDM is depicted. The model consists of two parts. The first part is the encoder which encodes the bag of words representation into a latent vector. The latent vector is sampled from a Gaussian with $\mu = 0$ and $\sigma = 1$. The second part is the decoder which translates the continuous latent representation to a bag of words using an MLP with a softmax as the last layer.

Apart from the way the posterior is estimated, NVDM is different from LDA in two ways. firstly, there is no explicit Topic-Document table, this makes the model less interpretable but allows for more complex relations between the topics and the words. To still find the most probable words one can fix the latent vector on a single topic and look at the softmax probabilities. Secondly, LDA uses a Dirichlet prior over the topic distribution where NVDM uses a Gaussian prior.

The model achieves lower perplexity results than LDA on most datasets, however, the topics seem less interpretable (A. Srivastava and Sutton, 2017). The reason to explore this topic model is because it is the simplest topic model using a VAE which allows us to use it as a baseline for the neural topic methods.

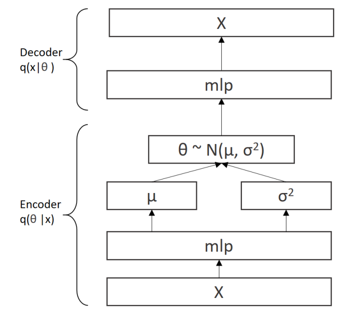


Figure 3.6: NVDM (Miao, Yu, and Blunsom, 2016)

3.3.3 Prodlda

Prodlda can be described as LDA where the only difference is that the topic-word mixtures are not constrained by a multinomial distribution but exist in natural parameter space. This means that β is unnormalized and the conditional distribution of words is given by $w_n|\beta, \theta \sim$ Multinomial(1, $\theta(\beta\theta)$). Although theoretically not required, neural variational inference (NVI) is used to approximate the posterior, just like NVDM (A. Srivastava and Sutton, 2017). This is why we will refer to Prodlda as a neural topic model.

To impose the Dirichlet prior on the topic distribution a softmax is used over the latent space θ. A schematic representation of this can be found in figure 3.7.

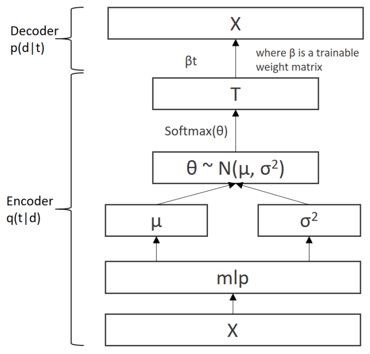


Figure 3.7: Prodlda (A. Srivastava and Sutton, 2017)

The advantage of Prodlda is twofold. It scales well to many documents because NVI can be parallelized. On top of this it also finds more coherent topics than LDA (A. Srivastava and Sutton, 2017). It should be noted that the perplexity, a metric that is often used to compare topic models, is higher than that of LDA. However, this is probably not the best way to compare topic models since it only gives information on the predictive capacity of the model rather than the quality of the topics we are interested in (Chang et al., 2009).

3.3.4 Gaussian Softmax Model

Lastly, we will have a look at the Gaussian Softmax model GSM introduced by Miao et Al in 2017. This model is similar to Prodlda except for two components. Firstly, there is a linear layer between the sampled continuous vector θ and the softmax layer that normalizes the topic distribution. Thus $t = softmax(w^T\theta)$. Secondly, β is also normalized such that it follows a multinomial distribution. To still create a model that can capture complexity between the topic vector and the reconstruction layer, an additional linear layer is added between the topic layer and the layer that represents β. For a proper overview we suggest to take a look at figure 3.8 which depicts the GSM model.

Miao et Al. (2017) show that the topic model outperforms NVDM and LDA in perplexity. On top of this Wang et Al. (2017) show that the model performs relatively well on topic coherence. Unfortunately they do not mention Prodlda in their comparison, so we do not know whether GSM outperforms Prodlda on topic coherence.

But what might be the most important reason for exploring this model is that both Miao et Al. (2017) and Wang et AL. (2017) have attached a language model to this topic model which is exactly what we intend to do in the next chapter.

Unfortunately all our experiments with the GSM model were unsuccessful. Our implementation based on the paper produced a very skewed distribution in the latent space leading to near random results.

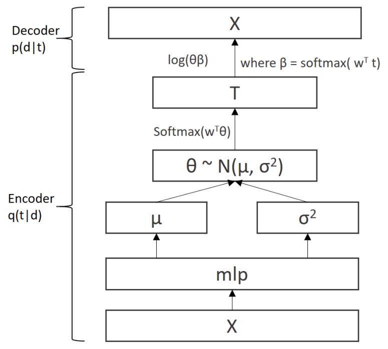
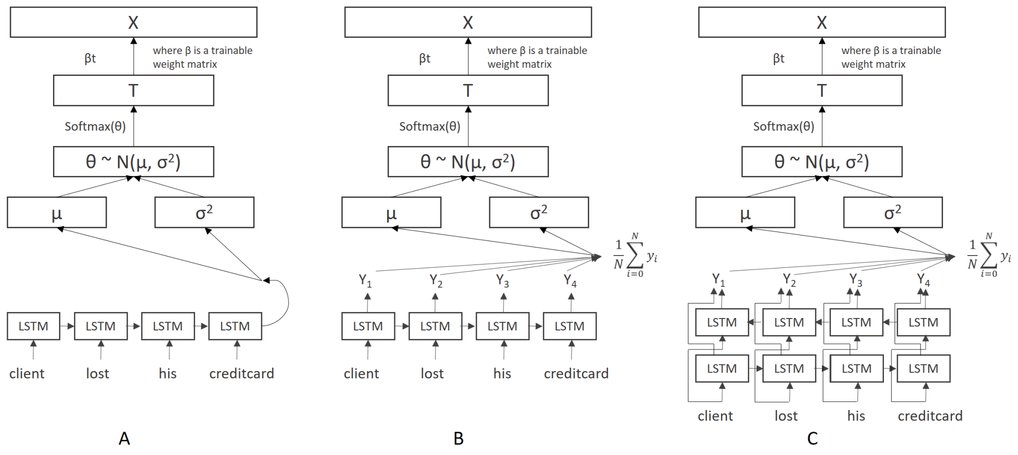


Figure 3.8: Gaussian Softmax Model (Miao, Grefenstette, and Blunsom, 2017)

3.3.5 Prodlda with LSTM encoder



X
βt
where β is a trainable weight matrix
T
Softmax(θ)
θ ~ N(μ, σ²)
μ
σ²
$\frac{1}{N}\sum_{i=0}^{N} y_i$
Y1
Y2
Y3
Y4
LSTM
client
lost
his
creditcard
A
B
C

Figure 3.9: Prodlda with an LSTM as an encoder instead of a MLP. A) uses the hidden state to encode the sentence. B) uses the average of the outputs to represent the sentence. C) similar to B except that it employs a bi-lstm)

The main shortcoming with the models discussed so far is that the document is encoded as a bag of words which does not take word order into account. Since the documents in the ABN AMRO dataset are rather short, the extra information in the word order can be crucial to find the proper semantic meaning. For this reason we propose a model that uses an long short-term memory (LSTM) based recurrent neural network (RNN) as an encoder for the Prodlda model instead of a normal MLP layer. This is not the first time somebody suggests such a model. Miao et Al (2016) already mention that the MLP in a Neural topic model can be replaced by an LSTM or a convolutional neural network (CNN). However, to the best of our knowledge results of such a model have not been published yet. We propose three ways to encode the documents using an LSTM which are depicted in Figure 3.9.

The first encoder we propose uses the last hidden state of the LSTM to represent the document. This is similar to the approach Bowman uses for his recurrent VAE (Bowman et al., 2015). This recurrent VAE encodes sentences in latent space using a LSTM as encoder and an LSTM as decoder with a latent layer in between. He showed that by using this method he was able to create a semantically meaningful latent space.

The second encoder we propose uses the average of the outputs of the LSTM. In this way we get information from everywhere inside the LSTM and not only at the very end.

The third encoder we propose is similar to the second, it uses the average of the concatenated output of a bidirectional LSTM (bi-LSTM). The latter is simply two LSTMs that are going in opposite directions over the input. This method of encoding the semantic information has been successfully done before, for example in (Rios, Aziz, and Sima'an, 2018). The advantage of bi-LSTM is that its output has knowledge of both sides of its surroundings rather than one. If we take the sentence *"This guy is crazy fun"* as an example. An LSTM trained to find sentiment would probably give a negative sentiment after four words. Because the bi-LSTM also has knowledge about the last word of the sentence it is able to assign the right output throughout the sentence, which in this case should be a positive sentiment.

We foresee that the third encoder will be most successfully because it provides most information to the output. Nevertheless, by trying all three we can see how much impact the individual choices have on the encoder (e.g. LSTM vs bi-LSTM).

3.4 Experimental set-up

In this section we try to answer the third research question, *"What topic model is the most suitable choice to distinguish different types of questions?"*, by testing several topic models on both public datasets as well as the ABN AMRO dataset.

All models, except for LDA, are implemented in Python (Van Rossum, 1995) in combination with Tensorflow (Martín Abadi et al., 2015). The experiments for LDA were also performed in python but we used the lda implementation from Scikit-Learn instead of implementing it ourselves (Pedregosa et al., 2011). For NVDM and Prodlda Tensorflow implementations from the authors were available on Github and we based parts of our code on their implementation.

3.4.1 Data

We will use three datasets to test the topic models: *20NewsGroups*, *APnews* and the ABN AMRO dataset. *20NewsGroups* is probably the most common dataset to test topic models. It consists of 18,000 Newsgroup posts on 20 topics. *APnews* is a collection of Associated Press news articles from 2009 to 2016, it's a about 15 times bigger than *20Newsgroup* which might give the neural topic models an advantage. Lastly, we will test the model on the ABN AMRO data. This is the most important test because it is the only data that contains customer inquiries which is what we are interested in. A table of summary statistics can be found in Table 3.1

	20NewsGroups	APNews	ABN AMRO
#docs	11,314	164,465	938,459
average doc length	11	20	24
max doc length	914	179	2,000
min doc length	0	0	1
vocabulary	7,357	12,112	4,222

Table 3.1: Summary statistics for data used in experiments

During preprocessing we tokenise words using the NLTK tokenizer (Bird and Klein, 2009), and lowercase all words. Additionally we filter low/high frequency words and stopwords using the TF-IDF vectorizer from the Scikit-learn package (Pedregosa et al., 2011). Words that occur in less than 10 documents or more than 95% of the documents are removed. Because we use the preprocessed 20NewsGroups data by Scikit-Learn we unfortunately only have 11096 documents for the 20Newsgroups data instead of the earlier mentionded 18.000. Nevertheless, this also removes headers and footer which should result in better performance because there is less noise in the data.

3.4.2 Evaluation

We evaluate our topic models in four ways: perplexity, recall, topic coherence and by qualitatively evaluating the quality of the topics found. For all our models we use a train, validation and test set. The test score we present is based on the model with the lowest perplexity on the validation set.

Perplexity

The most common method to evaluate language models is perplexity Chang et al., 2009. This metric is a measurement of how well a probability model predicts a sample. Often it is explained as the metric which indicates how 'perplexed' the model is by seeing the data. The higher the perplexity the more it is 'perplexed' to see the data. This means the lower the perplexity the better the model is able to predict the data. In our experiments we will both look at the train and test perplexity which gives us not only insight in how well it predicts the data but also whether it overfits.

To calculate the perplexity we follow Miao, Yu, and Blunsom, 2016 using:

$$exp(-\frac{1}{D}\sum_{d}^{D}\frac{1}{N_d}\log p(X_d)) \tag{3.2}$$

Where D is the number of documents, N_d is the number of words in the document and log p(X) is the log probability of the words in the document.

Recall@n

To have a more intuitive measure of the predictive capacity of the model we also look at the recall of the model. This means that we look what fraction of the n most probable words returned by the document are present in the document. We will indicate how many of the most probable words we take into account by naming the measure recall@n, where n can be any number smaller than the vocabulary size.

Topic coherence

Perplexity and recall are focused on the predictive probability of the model, however, what we are actual interested in is the quality of the topics in the model. Lau et Al (2014) give a comparative review of automatic topic coherence methods and shows that Normalized Pointwise Mutual Information (NPMI) is the metric that correlates most with human judgment. For this reason we will use NPMI in our thesis as a objective measure for topic coherence. NPMI can be calculated using:

$$NPMI(w_i) = \sum_{j}^{N-1}\frac{log\frac{p(w_i,w_j)}{p(w_i)p(w_j)}}{-logP(w_i,w_j)} \tag{3.3}$$

In figure 3.10 you find an example of topic coherence per sentence to get a feeling how the numbers relate to the topics.

We use the implementation by Lau et al. (2014) to calculate the topic coherence.

coherence	topic
0.94	company sell corporation own acquire purchase buy business sale owner
0.91	age population household female family census live average median income
0.86	jewish israel jew israeli jerusalem rabbi hebrew palestinian palestine holocaust
0.85	ship navy naval fleet sail vessel crew sink sea submarine
0.83	guitar album track bass drum vocal vocals release personnel list
...	
0.35	number code type section key table block set example oth
0.34	group official website member site base oth form consist profile
0.34	part form become history change present create merge forme separate
0.29	know call name several refer oth hunter hunt thompson include
0.27	mark paul heart take read harrison follow become know include

Figure 3.10: An example of topic coherence per topic from Lau et Al (2014).

	Newsgroups			*APNews*			*ABN AMRO*		
#topics	10	20	50	10	20	50	10	20	50
Perplexity	422	371	**328**	549	501	**494**	358	330	**263**
Recall@3	**0.29**	0.28	**0.29**	0.68	0.70	**0.72**	0.37	0.37	**0.38**
Topic Coherence	**0.12**	0.10	0.05	0.09	**0.10**	0.06	0.07	**0.09**	0.08

Table 3.2: Overview of quantitative results LDA

Qualitative approach

Although perplexity and topic coherence give us an objective indication of how well the models work, in essence they are both merely proxies to objectively and cheaply determine the quality of the topic models. Since, the most likely words given the topic $p(w|t)$ are used to infer the topics. In our qualitative evaluation we will therefore look at the 10 most likely words given the topics to determine the quality of the topic model.

Ideally we would have a team that rate the quality of the topics to determine which models deliver the best topics, this however is not possible due to time constraints. Nevertheless, we can use our own judgment to evaluate the topics and use this as an additional check on top of the objective metrics.

3.5 Results

In this section we will share our findings. For most results we will use a sample of 10k documents for computational reasons. The data-set is split in 60% training, 20% validation and 20% test. The neural models are trained on a GPU to speed up computation. For the public datasets this is done on a GTX 1070 with 8gb memory, for the abn-amro dataset this is done on the Laptop GPU which is a Nvidia Quadro M2000 with 4gb memory.

3.5.1 LDA

LDA will be used as the baseline to compare the other methods by. To assure the quality of the LDA model we use LDA model from scikit-learn with the default parameters. In table 3.2 you find an overview of the quantitative LDA results. What we observe is that the perplexities change relatively much between different datasets. The perplexity is lowest for the ABN AMRO dataset which can indicate that the documents are more uniform and therefore easier to predict than the other datasets. Recall@3 also differs per dataset and seems correlated with the perplexity. The latter makes sense because recall and perplexity both measure the predictive performance. Maybe most interesting is that the Topic coherence seems to be negatively correlated with the Perplexity for these three datasets. Obviously, an untrained topic model wouldn't yield any topic coherence, however, these results prove that lower perplexity doesn't necessarily lead to higher topic coherence.

One of the most important parameters in a topic model is the number of topics. We experiment with three topic sizes: 10, 20 and 50 topics. In table 3.2 one can observe that perplexity seems to decrease when the number of topics increase. This makes sense because the model has more parameters to fit the model to the data. It seems that the perplexity can still be improved by using more topics, however, this seems to have a negative effect on the topic coherence which is the measure we most care about because of its correlation with human judgment.

Obviously the number of topics is not merely one of the parameters one wishes to tune. It is also a preference. Maybe one wants to only find the most general topics and chooses to only have 10 topics. Others might want to have a more detailed look at the topics and choose to have 50 topics. For this reason we do not treat the number of topics as tuning and display the test scores for 10, 20 and 50 topics; similar to Strivastava et Al. (2017) and Miao et Al. (2016). This allows us to tune the models on validation perplexity and display the performance on an unseen test set for all three topic configurations.

Now let us have a look at the topics that the model found. In table 3.3 you find a sample of topics found by LDA in the 20Newsgroups dataset with their corresponding topic coherence. We can observe that it finds some sensible topics. Such as topics about war and religion. We also see topics that don't resemble any clear topic like the one in the very first row. From now on we will therefore make the distinction between *interpretable topics*, which are topics that resemble a topic interpretable by humans and *artificial topics* which are not interpretable but only exist to improve predictive performance of the model.

#topics	topic coherence	sampled topic
10	0.07	10 high year 1993 car 000 years cost power 100
	0.33	14 11 24 16 13 15 10 25 17 18
	0.14	god believe people does christian jesus bible say christians life
	0.22	game team games season win league year player tv hockey
20	0.21	war turkish armenian armenians army jewish argic serdar soldiers killed
	0.30	bike miles rear oil tires seat harley valve fork springs
	0.04	pl mn mx s2 lf tt wm mr sl ax
	0.25	10 edu 15 1993 11 12 20 30 18 13
50	0.00	org dod wide email 11 appropriate com good just official
	0.05	don like just think people know lot probably use really
	0.14	space research information development nasa internet program data science university
	0.36	15 11 10 16 13 20 14 12 18 17

Table 3.3: Sampled LDA topics on 20Newsgroups

#topics	topic coherence	sampled topic
10	0.11	woning notaris krediet hyp direct hypotheek cli ivm klant nt
	0.21	nummer number gesproken bc telefoonnummer datum opdracht graag telefonie uitgevoerd
	0.04	number 2017 eur bedrag code 00 mut nr 17 klant
20	0.10	number graag groet vriendelijke heer ontvangen telefoonnummer amro nummer bc
	0.37	abn amro to the bank on for and please card
	0.19	bedrag number euro gestort geld bijgeschreven automaat storing storten klachtomschrijving
50	0.45	to the on for and please account your amro abn
	0.02	vraagt post client gedaan tevens betaald vind bestellen staat meneer
	0.21	nummer number bc gesproken opdracht datum telefoonnummer telefonie graag uitgevoerd

Table 3.4: Sampled LDA topics on ABN data

Another thing we notice is that the topic coherence seems to correspond with what we view as *interpretable* topics which confirms that we can use topic coherence as a proxy for quality. There is also one topic that resembles numbers. This one is interesting since a human would probably not pick this as a topic. On the other hand one immediately sees the coherence between the 'words' in the topic which legitimizes the high topic coherence.

If we compare the quality of the topics with respect to the number of topics in the model we would expect the model to find more detailed topics when we have more topics. For example, that the topic 'education' would be split in several topics like 'teachers', 'students' and 'classrooms'. However, it seems that when the number of topics increase mainly the artificial topics increase. We also see that there is often more overlap between the topics.

In Table 3.4 you find a sample of topics found by LDA in the ABN AMRO data. It is interesting to see that it either finds Dutch or English topics. Before looking at the topics we weren't even aware of the English documents in the dataset but the topic model was able to distinguish the languages. We see that LDA finds meaningful topics in the ABN AMRO data, like the one on the very first row. This topic clearly resembles questions regarding mortgages for a house. However, even though it is a clear topic it doesn't provide information which the bank can leverage to reduce this kind of questions. We will elaborate on this problem and with possible solutions in future sections. Just like with the 20Newsgroups data it finds a lot of 'artificial topics' which also are not useful to find the most frequently posed questions.

All experimental results have been done using only the 1000 most important words from the vocabulary in the model. We did this because we initially observed that the larger vocabulary increased the perplexity. In the case of ABN AMRO with 20 topics using all words as described in the experimental set-up the perplexity would increase from 344 to 964. However, later we found a larger vocabulary has a positive impact on the topic coherence. In this same example the topic coherence would go from 0.09 to 0.12.

To conclude, like expected LDA is able to find meaningful topics and is therefore suitable to be used as a baseline.

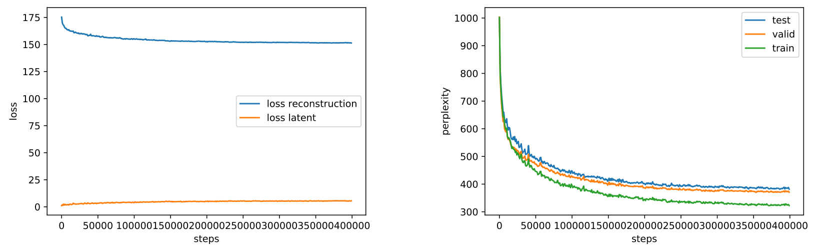


Figure 3.11: NVDM training progression on 20Newsgroups with 10 topics.

	Newsgroups			*APNews*			*ABN AMRO*		
#topics	10	20	50	10	20	50	10	20	50
Perplexity	404	**379**	395	481	**457**	461	238	196	**179**
Recall@3	0.31	**0.34**	**0.34**	0.78	0.78	**0.80**	0.47	**0.51**	**0.51**
Topic Coherence	0.04	0.04	**0.04**	0.01	**0.01**	0.01	0.01	0.01	0.01

Table 3.5: Overview of quantitative NVDM results

3.5.2 NVDM

Determining the performance of a neural topic such as the NVDM model is a little more tricky. This is because there are a lot of ways to tune the model. Besides number of topics we can play with the number of hidden units, the activation functions, the weight initialization and the optimizer. For our model we use a combination of best practices and what the authors mention in the paper as their best parameters. We initialize the weights using the Xavier distribution, use the Relu activation function and optimize using Adam for the reasons mentioned in the technical background. The number of hidden units is set to 500 hidden units which is equal to that of the original paper Miao, Yu, and Blunsom, 2016.

To determine the best model we save the model only when the perplexity on the validation set is lower than all earlier results on the validation set. In this way we avoid choosing a model that overfits on the training set. Unfortunately this means that we sometimes pick a model that has a good performance on the validation set but poor performance on the test set. In the case of NVDM with 10 topics this means that the perplexity on the train set is 350 on the train set, 388 on the validation set, but 404 on the test set. Nevertheless, this is the most reliable method for finding a model that generalizes well.

In figure 3.11 you find how the loss and perplexity progresses during training. The loss is composed of the reconstruction loss and the latent loss (KL term). The reconstruction loss gradually decreases during training whereas the latent loss the seems to increase slightly. This is normal behavior for a VAE and is caused because the errors that are propagated backwards from the reconstruction loss are stronger than that of the KL loss. The perplexity also gradually decreases which confirms that the model is learning to reconstruct the data. The model seem to perform a lot better on the train set, however improvements in the train set don't seem to have negative effects on the validation set. Although this data doesn't tell us much about the performance, it indicates that the model is correctly implemented.

In table 3.5 the quantitative results of NVDM are depicted. We observe that NVDM performs better in recall and perplexity than LDA. For 20Newsgroups and APnews predictive performance seems to go down with 50 topics based on the perplexity results. This is probably caused by overfitting, for this reason regularization such as l2 or dropout might be useful when one wants to use this model with more than 20 topics. Although the perplexity is high, the topic coherence is much lower than with LDA for all datasets.

The low topic coherence is confirmed when we have a look at the topics in table 3.6. The most likely words given the topic often overlap and we cannot really find *interpretable topics*. This indicates that this model is not useful as a topic model since the latent space only consist of *artificial topics* that only aid the predictive performance of the model.

From these results we can conclude that NVDM works well as an auto-encoder, however, we would not advice to use it as a topic model because it has a low topic coherence and no *interpretable topics*.

#topics	topic coherence	sampled topic
10	0.04	like just time new know don does people think make
	0.03	like just know don time think does people ve use
	0.04	know like just does don think people make new time
20	0.03	like just time new know don does use think people
	0.05	like just know don does time think good use make
	0.04	like just know don does time think use good way
50	0.05	like just know don think time does good use make
	0.04	like just know don does think use make time new
	0.03	like just don think know time good new make ve

Table 3.6: Sampled NVDM topics from 20Newsgroups

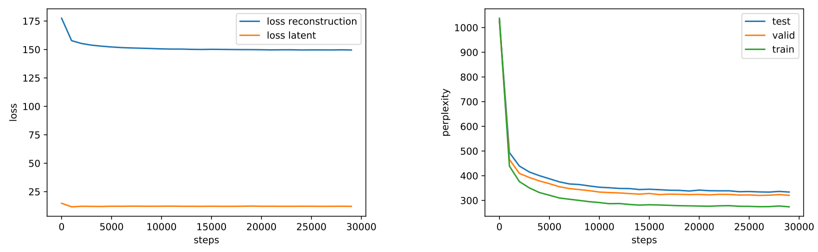


Figure 3.12: prodlda loss and perplexity progression for 20Newsgroups with 10 topics

3.5.3 Prodlda

The second neural based topic model is Prodlda. Similar to NVDM we used Xavier initialization, ReLu activations and the Adam optimizer. The number of hidden units we used was only 100, the same number as used in the original paper (A. Srivastava and Sutton, 2017).

In figure 3.12 the progression of the loss and perplexity are depicted for 20Newsgroups dataset with 10 topics. Here we observe how both the reconstruction and KL loss decrease. However, even though the latent loss increased during training for NVDM, the latent loss after convergence is still higher for Prodlda with 7.1 against 4.5 for NVDM. If we take a look at the perplexity progression we see that it decreases more rapidly and smoother than NVDM which indicates that the loss space is smoother as well.

In table 3.7 we find an overview of all quantitative results for Prodlda. We observe that it has in all cases higher recall than both LDA and NVDM. Also the perplexity is lower than LDA in all cases and in some cases even lower than NVDM. This indicates that the model has a solid predictive performance. On top of this the model has higher topic coherence than NVDM that is slightly lower than LDA. This is interesting because A. Srivastava and Sutton, 2017 report that Prodlda should outperform LDA regarding topic coherence. In retrospect this might have to do with the smaller vocabulary size that we are using.

In table 3.8 and 3.9 sampled topics are depicted for both 20Newsgroups and the ABN AMRO data. We observe that is has similar topics as LDA. For example it also has a topic containing numbers that is slightly polluted with other words. Compared to LDA the topics seem to be a little less coherent which is backed up by the topic coherence measure.

All things considered, Prodlda seems to be a better topic model than NVDM because the topics returned by Prodlda are interpretable whereas NVDM's topics are not. Even though the perplexity of Prodlda is much lower than LDA, LDA still seems to outperform Prodlda because the topics returned by LDA are more coherent. However, for large datasets Prodlda could still be a more appropriate model because it scales better.

3.5.4 Prodlda with LSTM encoder

Lastly we will have a look at our three proposed LSTM encoders for Prodlda. For these experiments we take 20 topics because we observed that in previous methods this yielded the best topic coherence. Also because the ABN AMRO dataset is the only dataset that has smaller documents we will only test the model on this dataset. For all encoders we use a total of 200 hidden units. For the bi-lstm we use only 100 hidden units per layer such

	Newsgroups			*APNews*			*ABN AMRO*		
#topics	10	20	50	10	20	50	10	20	50
Perplexity	404	329	**236**	513	458	**397**	298	241	**184**
Recall@3	0.35	0.41	**0.56**	0.75	0.83	**0.83**	0.43	0.5	**0.6**
Topic Coherence	0.06	0.07	**0.09**	0.04	0.04	**0.04**	0.03	**0.05**	**0.05**

Table 3.7: Overview of quantitative Prodlda results

#topics	topic coherence	sampled topic
10	0 0.06	does god people point don say just believe think know
	0.07	think just good game time play like year don team
	0.16	20 com 10 end 14 15 16 new 30 11
20	0.12	thanks looking help know information does advance hi appreciated anybody
	0.07	just like car little bike think good right road cars
	0.16	god jesus bible christians does believe faith say people christian
50	0.05	soon pain blood medical human disease usually edu health experienc
	0.09	window event widget use application following trying does events problem
	0.09	tv heard local hear radio seen night news games ve

Table 3.8: Sampled Prodlda topics on 20Newsgroups

#topics	topic coherence	sampled topic
10	0.04	klant ontvangen zie gebeld number brief verzoek contact meneer geeft
	0.08	number gesproken klant graag 2017 nummer bc rekening telefoonnummer datum
	0.00	belt kantoor gaat meneer number klant rh mevrouw nieuwe staat
20	0.06	rekening rh belt meneer ot geeft staat ib gaat veri
	0.09	number graag 2017 amro abn rekening bank mevrouw heer nummer
	0.02	rekening number kosten meneer klant gaat extra bedrag geopend nieuwe
50	0.38	number abn bank amro please your the have for you
	0.07	overboeking number belastingdienst betaling overgemaakt klant afgekeurd gedaan gelukt afge
	0.02	number klant meneer gaat wilde bedrag weten 20 vraag 00

Table 3.9: Sampled Prodlda topics on ABN AMRO dataset

	LDA	*Prod*	*prod-LSTM-h*	*prod-LSTM-m*	*prod-LSTM-m-bi*
#topics	20	20	20	20	20
Perplexity	303	**209**	512	293	273
Recall@3	0.28	**0.41**	0.27	0.30	0.34
Topic Coherence	**0.10**	0.05	0.04	0.08	0.09

Table 3.10: Overview of quantitative Prodlda with LSTM encoder results on ABN AMRO data with only documents of <50 words.

that performance is not affected by the number of trainable parameters. We first did all experiments on a new ABN AMRO dataset for which we removed all documents larger than 50 words. In reaction of the observed results we then also did experiments on a dataset where we removed all documents larger than 30 words. To fairly compare the results of this new model with the previous models we ran these models also on these new datasets. We didn't include NVDM in these results because it was clearly not performing as well as LDA and Prodlda

In table 3.10 one finds the results for the ABN AMRO data with documents not larger than 50 words for LDA, Prodlda, Prodlda with lstm encoder that uses the hidden state, Prodlda with lstm encoder that uses the mean of the outputs, and ProdLDa with bi-lstm encoder that uses the mean of the outputs. The first thing that we observe is that lda still has the highest topic coherence and that Prodlda still has the highest perplexity and recall. But before we discard them, let's first have a closer look at their results.

The worst performing encoder is the one that uses the hidden state of the LSTM. This one has the highest perplexity, and lowest recall and topic coherence of all models. When we use the mean of the outputs instead of the hidden state the performance shoots up on all fronts. Especially the topic coherence that goes from 0.04 to 0.08 is interesting because this means that this topic model is able to achieve higher topic coherence than Prodlda on datasets with short documents. Lastly we look at using the mean of the bi-lstm instead of only a one directional lstm. This means that the output is also aware of the text ahead in the document. This addition also increases the performance on all fronts.

Because of the promising results of the bi-LSTM encoder, we are interested in how it compares to the other topic models when documents are even shorter, a situation where the LSTM encoder should have an advantage. For this reason we created a dataset with only sentences shorter than 30 words, the results on this dataset are depicted in figure 3.11. As expected, Prodlda with bi-LSTM encoder now acquires the highest topic coherence which confirms our idea that the local information is of added value when a topic model has to deal with short documents. LDA on the other hand suddenly has a drop in topic coherence performance which gives it the lowest topic coherence. On top of this, its perplexity is lowest in comparison to the other models. These remarkable results for LDA might be caused by the higher variance between the validation set and test set. Because less words are used per document it makes sense that the average Euclidean distance between bag of word distances is larger. This is confirmed when we look at the perplexities for Prodlda where train ppl is 121, valid ppl is 199 and test ppl 252. In this case the test perplexity gives a distorted image and a larger validation and test set should be considered for more robust results.

In figure 3.11 3 random topics from Prodlda with the bi-LSTM encoder are depicted. At a first glance the topics seem coherent, probably also because the documents contain less redundant information which potentially pollute the topics.

From these experiments we can conclude that if one wants to attach a LSTM as an encoder to a topic model it is best to use a the outputs of a bi-LSTM rather than the hidden state. On top of this we find that Prodlda with bi-LSTM encoder might be your preferred choice when dealing with documents shorter than 30 words because in this specific scenario it will return the most coherent topics.

#topics	topic coherence	sampled topic
20	0.18	nummer bedrag abn klachtomschrijving gestort transactiedatum email amro gehele
	0.08	aangegeven betaalpas volgende verkopen relatie zakelijke aav beeindigen legitimatie week
	0.06	ivm gebruik notaris loopt woning verkoop fk afspraak number overlijden

Figure 3.13: Sampled topics from Prodlda with bi-LSTM encoder on ABN AMRO dataset

	LDA	*Prod*	*prod-LSTM-m-bi*
#topics	20	20	20
Perplexity	**239**	252	253
Recall@3	0.26	**0.30**	**0.30**
Topic Coherence	0.06	0.08	**0.09**

Table 3.11: Overview of quantitative Prodlda with LSTM encoder results on ABN AMRO data with only documents of <30 words.

3.6 Conclusion & Discussion

In this chapter we aim to answer two questions: *"Is as topic model a suitable choice to distinguish different types of questions?"* and *"What topic model is the most suitable choice to distinguish different types of questions?"*. The first question was discussed in the section - what is a topic model? - here we concluded that topic models are a suitable choice to distinguish different types of questions if one doesn't mind a few redundant topics. This conclusion was confirmed in the result section in which the topic models were able to find topics in all three datasets including the ABN AMRO dataset that contains questions by customers.

To determine which topic model is most suitable to distinguish different types of questions we looked at LDA, NVDM, Prodlda and our proposed Prodlda with bi-LSTM encoder. We evaluated them using perplexity, recall and topic coherence. Because topic coherence has shown to correlate most with human judgment this is in our opinion the most important metric. From our results we can say that in most cases LDA seems the best topic model because of its high topic coherence for all datasets. When the documents are smaller than 30 words we have shown that Prodlda with bi-LSTM can outperform LDA on topic coherence and one can therefore consider this model for datasets with small documents. Since questions are often short this might therefore be the best model to distinguish different types of questions.

This however is not the complete story. Prodlda also had good results and according to Strivastava (2017) it should outperform LDA on topic coherence. In our experiments this was not the case which could be because our model was not tuned optimally. For our results we only used best practices and recommended sizes which don't guarantee optimal results for our datasets. Nevertheless, this stringent dependency on tuning for neural networks should in general also be considered as a practical disadvantage on its own. This might not be a hurdle for researchers that have a lot of computational power, but for the average data scientist it is. Despite this, Prodlda almost performed as well as LDA, and it should be noted that the parallelizable nature of neural topic models gives it a leg up when working with larger data sets.

Chapter 4

Language Model

Language modeling is a key component in the NLP Toolkit. It allows the user to determine the probability of a sequence given a language. The ability to determine the probability of a sequence is valuable to increase the performance of various applications, such as: machine translation (Schwenk, Rousseau, and Attik, 2012), speech recognition (Mikolov et al., 2010) and handwriting recognition (Plotz and Fink, 2009). It can also be used to determine the most likely sequence which has also found its way in many applications. For example: sentence completion (Gubbins and Vlachos, 2013), spelling correction (Kernighan, Church, and Gale, 1990), and it is even used to write entire books (Velzen, 2017).

In this thesis we will focus on the latter because we use a language model to create sentences that resemble a topic from a topic model. It therefore tries to find the most probable sequence of words given the corpus and the topic. We do this by conditioning a model on the previous words and the topic distribution. In the next sections we will elaborate on the history of language models, the current state of the art models and how to condition neural language models.

4.1 What is a language model?

Within computational linguistics a language model is defined as *the probability distribution of a sequence of words given the language* (Bengio et al., 2003). This can either be a small sequence, a single word, or an entire document. Suppose there is a sequence $\mathbf{w} = (w_1, ..., w_N)$, where w represents a word and N is the number of words in the sequence. A language model would then ideally be able to find $p(\mathbf{w})$ of the true language distribution. Since the true distribution of *the English language* isn't a real thing, the language model determines the distribution using a fixed dataset. When applying the language model we hope that the used dataset, which is a sample of the full language, resembles the *true* distribution of the language. For this reason scientists have been using increasingly large datasets to train their language models, the hope is that the larger the dataset, the closer it resembles the *true* language distribution (Chelba et al., 2013). The language model in itself is a mathematical formula that is regulated by a set of free parameters. The probability distribution the model outputs is dependent on these free parameters. How these parameters are trained depends on the language model.

Although we call it a language model and focus on language, these models can be used to calculate the probability of any type of sequence given a corpus. And even if we talk about language, it's possible that we only want the model to represent a subset of the language. For example, doctors might want different sentence completion than police officers because their domain language is different.

In recent years the neural language models have been dominant because of their performance and ease of use. In the next section we will review the more traditional methods to give a better overview of the problem and what has been tried.

4.2 Historic context

The first statistical language model was proposed in 1980 to aid the performance of speech recognition (R. Rosenfeld, 1996). After its introduction it also found its way to statistical machine translation, information retrieval and many other applications. Due to this growing importance in many applications paramount of research has been done on many kind of language models. In this section we will go over the most influential models.

Most statistical language models decompose the probability of the sentence into a product of conditional probabilities. An example of this is depicted in equation 4.1. Here the probabilities of the words W given its history h is multiplied to find the full probability of the sentence. The main challenge for such models is to overcome the data sparsity problem that causes the probability of a sentence to go to zero if it hasn't seen the word or sequence of words in the dataset.

$$\begin{gathered} p(\mathbf{w}) = \prod_{i=1}^{N} p(w_i|h_i) \\ p(\textit{We love NLP}) = p(We)p(love|We)p(NLP|\textit{We Love}) \end{gathered} \tag{4.1}$$

N-grams used to be the most popular type of statistical model, which reduces the dimensionality of the history as a Markov source of order n-1. Thus equation 4.1 becomes equation 4.2, where n resembles the size of the n-gram.

$$p(\mathbf{w}) = \prod_{i=1}^{N} p(w_i|w_{i-n+1}, ..., w_{i-1}) \tag{4.2}$$

To determine the optimal value of n is a trade-off between the stability of the estimate and the accuracy of the estimate. Usually trigrams (n=3) are chosen when one works on large data-sets (millions of words) and bi-grams (n=2) are chosen when working with smaller datasets (R. Rosenfeld, 1996). However, even with this limited history the data remains sparse. In the Wall Street Journal that contains 38 million words, still one third of the tri-grams are only observed a single time. Since using this data directly would result in poor performance on unseen data because many sentences will get zero probability, many techniques have been developed to smooth the distribution to assign probability to unseen n-grams; Additive smoothing, Good-Turing smoothing, Katz smoothing and many more. The most competitive smoothing technique that is sometimes still used as a baseline against neural language model is modified Kneser-Ney smoothing (James, 2000). The key idea behind this model is to take some probability from observed n-grams to assign them to unseen n-grams. Also, when the frequency of a higher order n-gram is low or zero the model will determine the probability using the lower order n-grams, this concept is called *back-off*.

Although n-grams have been the most widely used and successful approach before neural models, many other paths have been tried to create a language model. Bahl et al., 1990 was the first to try a decision tree approach. This approach determines the next word in the sentence using the history in the sentence $p(w_i|h_i)$. At every node in the tree the most informative question about the history is asked to determine the next word in the sentence. It finds these most informative questions by looking which question yields the highest reduction in entropy. This approach was abandoned due to its low performance and high computational cost. Since a language model is about language, linguistically motivated approaches have also been tried. However, they were also abandoned because of low performance. Context free grammar was used to create a probabilistic model that was optimized by using Expectation-Maximization. The poor results were due to the many local maxima that were far away from the global which made it difficult to optimize (R. Rosenfeld, 1996). The models discussed so far all suffer from sparsity problems because the parameters are being determined with little data. Take the decision tree for an example, the deeper one goes into the tree the less data it has for the decision. Exponential models have been proposed to avoid this problem. In equation 4.3 the general form of this model is depicted.

$$p(\mathbf{w}|h) = \frac{\exp[\sum_i \lambda_i f_i(h,w)]}{Z(h)} \tag{4.3}$$

Z(h) is a normalizing factor over the history, $f_i(h, w)$ is an arbitrary function over the current word and history and λ_i is the trainable parameter. As one can observe is that even if a history doesn't exist the probability remains positive because they are summed instead of multiplied (Della Pietra et al., 1992). The model was relatively popular due to the elegance by which it handled data sparsity. The model performed similar to n-gram models with Kneser-Ney smoothing but due to the computational complexity of training these models, n-gram models became the standard for production level language models.

In 2003 the first neural based language model was introduced (Bengio et al., 2003). Even this first attempt already significantly improved the performance of the n-gram based models. This first proposed model was a multilayer perceptron (MLP) that predicts the probability of a word given an arbitrary number of words which is depicted in Figure 4.1. As one can see a concatenation of the word embedding of the previous words are used as input which goes through an MLP with tanh activation functions. The output, the probability of the word given the context, is determined by a softmax over the last layer that has the size of the vocabulary.

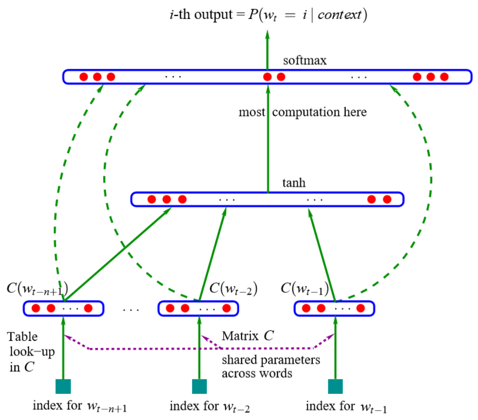


Figure 4.1: A schematic drawing of the first neural probabilistic language model as proposed by Bengio et al., 2003.

Since this first attempt a lot of research has been done to find the most effective neural language model. In the next section we will discuss the most successful neural language model approaches and why neural networks are so effective for language modeling.

4.3 Neural language models

Although the neural based approach was not popular from the start, over the years, it has become the standard in language modeling. Its success is embedded in the way it solves the sparsity problem in the data. Before the data is used by the model a word is converted to a continuous vector of a considerably smaller dimension than the vocabulary size, better known as a word embedding. At first this word embedding, or word feature vector as Bengio called them, was only perceived as a byproduct of the neural network. Later, a lot of interesting properties and uses have been found for these word embeddings and it has become a research field on itself.

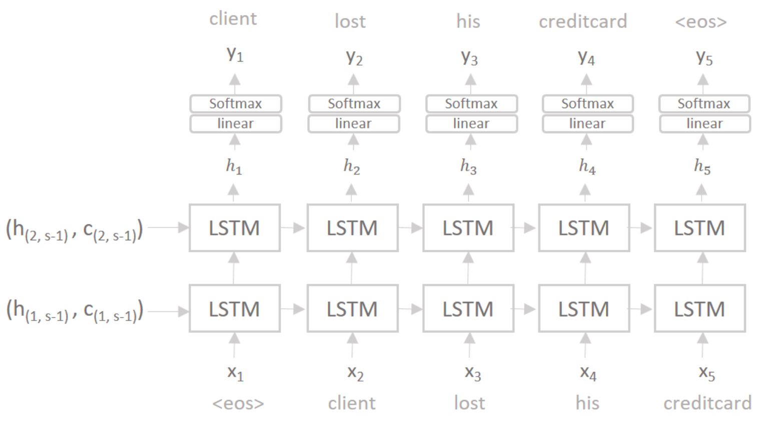


Figure 4.2: LSTM language model with two layers. The output of the LSTM is passed through a linear layer and a softmax to determine the most likely word.

A lot has happened since the introduction of Neural language models by Bengio in 2003 and most of the research has been focused on Recurrent Neural Networks (RNN). The advantage of this architecture is that it can, in theory, use all history of the previous words to determine the probability of the words with a fixed set of trainable parameters. Because the standard RNN cell isn't able to train long-term dependencies, better known as the vanishing gradient problem, Long Short-Term Memory (LSTM) have been used to attain stable results

(Hochreiter and Schmidhuber, 1997, Mikolov et al., 2010).

There are two main types of neural language models character based and word based (Jozefowicz et al., 2016). Word-level language models use individual words as their input which results in a large input and output dimension. This is partly resolved by removing rare words from the dataset and replacing them by `UNK` tokens. Character based language models have a smaller input and output dimension because they read the source character for character rather than per word. The advantage of character level models is that the input is less sparse, however, they tend to take longer to train and they can produce non existent words. We choose to use word-level language models for our experiments because it allows us to look at word probabilities for the language model in different situations.

In figure 4.4 you find the RNN architecture that we will use for our experiments. Every LSTM cell receives an input and a hidden state. Based on this information it will produce an output. A following layer can use this output as its input which gives the effect of multiple layers like an MLP. The output of the last layer will go through a linear layer and a Softmax to predict the next word in the sequence. Every cell also decides which information is important and passes this on as input for the next LSTM cell. The cell weights are shared across the cells in the horizontal dimension. In the past few years many LSTM cells have been proposed, in our model we will use the basic LSTM cell from Tensorflow which works as follows:

$$\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned} \tag{4.4}$$

x_t is the input, c_t is considered to be the cell state and h_t is used as output of the LSTM cell. Both c_t and h_t are passed to the next LSTM cell as a tuple and is considered the hidden state. f_t governs what needs to be forgotten in the cell state, i_t what needs to be added to the cell state and o_t what needs to be included in the output state h_t. All these factors contribute to a reusable block that where all information is captured in the weights.

To regularize the model a dropout layer is fitted between between the lstm layers and the output layer. This element is crucial to achieve low perplexities on unseen data (Zaremba, Sutskever, and Vinyals, 2014).

To train a language model efficiently on a text corpus the data is batched as depicted in figure 4.3. The corpus is represented as a sequence of letters in the alphabet. The corpus is first split up in the number of layers one desires after which one stacks them and divides them into batches. By using this method one can use the GPU optimally as well as being able to pass on the hidden state. The latter is done by saving the hidden state and using it as the initial state for the next batch. The very first state is initialized with zeros.

After training the model we can generate text from the model. This is done by initializing the hidden state with zeros and starting with an end of sentence token such that the model is aware it starts a new sentence. The predicted next word is then taken as input for the next time step to create sentences. The output can either be determined by greedily taking the highest Softmax probability, by sampling from the Softmax or by applying beam search over multiple time steps to determine the most likely sentence. In our examples we apply a greedy approach to generate our sentences.

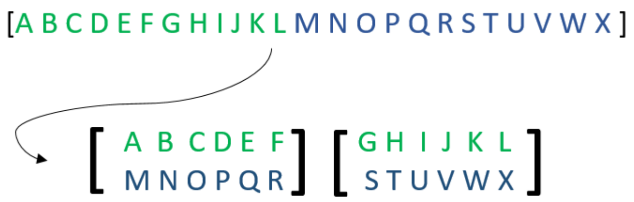


Figure 4.3: How a corpus is batched for a neural language model to make optimal use of the GPU. In this case every letter represents a word in a corpus.

4.4 Conditioned neural language models

To generate sentences that represent topics we need to adjust our language model such that it can take a conditional input during training and inference.

Mikolov (2012) are the first who condition a language model on a topic model. They do this to provide the language model with more global knowledge of the text such that the LM determines the probability of the next word, not only on the previous words, but on the entire context. By using the topic distribution based on the previous 40 words they were able to achieve a lower perplexity on the Penn Treebank dataset. They achieve this introducing a topic layer that both connects to the output and the hidden layer.

Lately other models have been proposed that jointly learn a neural topic and language model which goal it is to generate sentences to represent the topics. Adji B. Dieng and Paisley, 2016 introduce TopicRNN which combines a topic model with an LSTM by concatenating the topic of the document to the output of the LSTM. Nallapati, Zhai, and Zhou, 2017b introduce SenGen, short for Sentence generating variational neural topic model. This model jointly learns a neural topic model and LSTM language model by concatenating the topic vector to the input of the language model.

More recently, Wang et Al. (2018) proposed a more complex conditional LSTM. They also combine a neural topic model together with a conditioned language model. However, Instead of merely concatenating the topic vector to an input they create a Mixture of Experts language models that have separate weights assigned to each latent topic. Since using several language models is computationally inefficient they propose an LSTM cell that implicitly creates an ensemble of T language models, where T is the number of topics. This LSTM cell is defined as:

$$
\begin{aligned}
\tilde{x}_{*,t} =& W_{*,b}\mathbf{t} \circ W_{*,c}x_t \\
\tilde{h}_{*,t} =& U_{*,b}\mathbf{t} \circ U_{*,c}h_t \\
f_t =& \sigma_g(W_f \tilde{x}_{(f,t)} + U_f \tilde{h}_{(f,t-1)} + b_f) \\
i_t =& \sigma_g(W_i \tilde{x}_{(i,t)} + U_i \tilde{h}_{(i,t-1)} + b_i) \\
o_t =& \sigma_g(W_o \tilde{x}_{(o,t)} + U_o \tilde{h}_{(o,t-1)} + b_o) \\
\tilde{c}_t =& \sigma_c(W_c \tilde{x}_{(c,t)} + U_c \tilde{h}_{(c,t-1)} + b_c) \\
c_t =& f_t \circ c_{(t-1)} + i_t \circ \tilde{c}_t \\
h_t =& o_t \circ \sigma_h(c_t)
\end{aligned}
\tag{4.5}
$$

The conditional LSTM is similar to the regular LSTM except for the first two lines where a new inputs and hidden representations are created. These two lines are executed four times where we replace * by f, i, o and c. In the first line we multiply a batch of topic vectors and the batch of inputs by a separate weight matrix such that both results have similar dimensions. Now they have the same dimensions we apply the Hadamard product such that we create a new input for the LSTM that is conditioned on the topic model. In the second line the same is done for the hidden state. In figure 4.4 a graphical representation of this model is depicted which shows how the new language model is conditioned on topic vector t from both the hidden state and the input.

In our experiments we will try three methods to condition a language model on the topic distribution: the mixture of experts LSTM (LSTM-C-MoE), a LSTM conditioned at the input (LSTM-C-input) and a LSTM conditioned at the output (LSTM-C-output). LSTM-C-MoE is implemented as described in figure 4.4 and equation 4.5. LSTM-C-input is depicted in figure 4.5. Here the topic distribution of the document and the word embedding are concatenated and used as the LSTM's input. LSTM-C-output concatenates the topic distribution with the output of the LSTM which becomes the new input for output layer. The latter is depicted in figure 4.6.

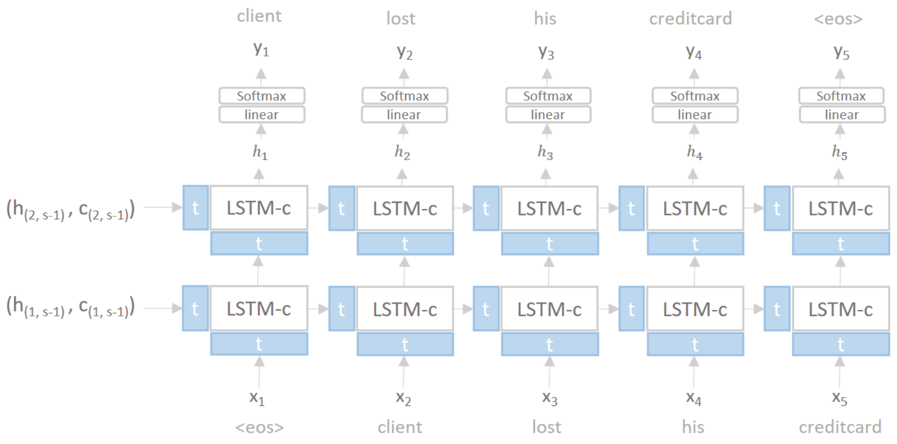
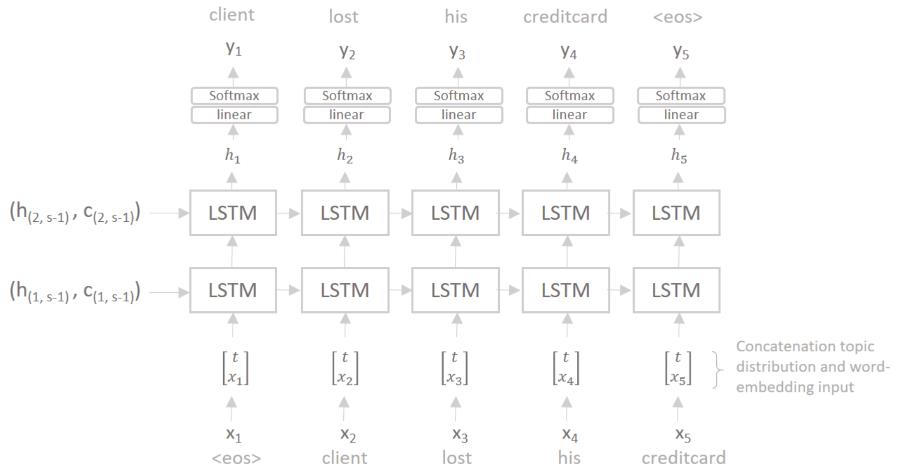
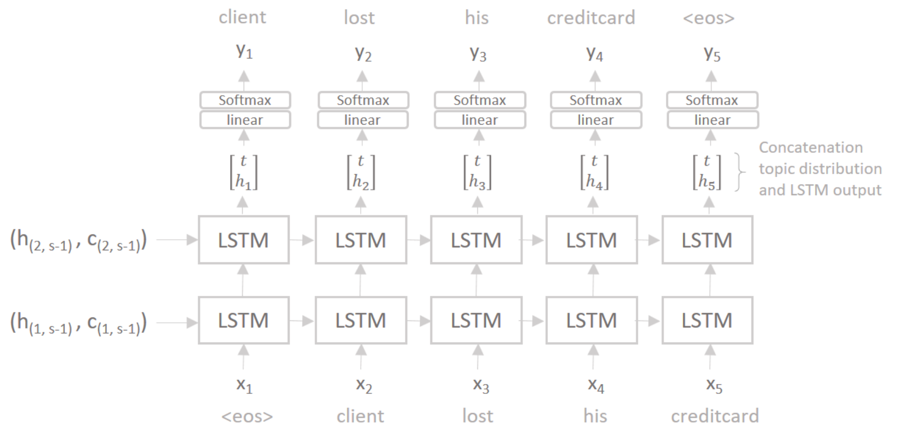


Figure 4.4: LSTM-C-MoE: Conditioned language model using special LSTM cells from Wang et al., 2017.



client
lost
his
creditcard
<eos>
y_1
y_2
y_3
y_4
y_5
Softmax
linear
h_1
h_2
h_3
h_4
h_5
$(h_{(2, s-1)}, c_{(2, s-1)})$
$(h_{(1, s-1)}, c_{(1, s-1)})$
LSTM
$\begin{bmatrix} t \\ x_1 \end{bmatrix}$
$\begin{bmatrix} t \\ x_2 \end{bmatrix}$
$\begin{bmatrix} t \\ x_3 \end{bmatrix}$
$\begin{bmatrix} t \\ x_4 \end{bmatrix}$
$\begin{bmatrix} t \\ x_5 \end{bmatrix}$
Concatenation topic distribution and word-embedding input
x_1
x_2
x_3
x_4
x_5
<eos>
client
lost
his
creditcard

Figure 4.5: LSTM-C-input: Language model conditioned on the topic distribution by using the concatenation of the word embedding and the topic distribution of the document as input to the LSTM.



client
lost
his
creditcard
<eos>
y_1
y_2
y_3
y_4
y_5
Softmax
linear
$\begin{bmatrix} t \\ h_1 \end{bmatrix}$
$\begin{bmatrix} t \\ h_2 \end{bmatrix}$
$\begin{bmatrix} t \\ h_3 \end{bmatrix}$
$\begin{bmatrix} t \\ h_4 \end{bmatrix}$
$\begin{bmatrix} t \\ h_5 \end{bmatrix}$
Concatenation topic distribution and LSTM output
$(h_{(2, s-1)}, c_{(2, s-1)})$
$(h_{(1, s-1)}, c_{(1, s-1)})$
LSTM
x_1
x_2
x_3
x_4
x_5
<eos>
client
lost
his
creditcard

Figure 4.6: LSTM-C-output: Language model conditioned on the topic distribution by using the concatenation of the LSTM output and the topic distribution of the document as input for the linear layer before the Softmax.

To create the topic distribution t we will use lda. This because in the previous chapter we established that LDA consistently scores high on topic coherence. In our implementation we first train lda on the train set after which we use this model to provide the topic vectors when training the LM. Although we first wanted to use the entire model from Wang et Al (2018), our implementation of their neural topic model, the Gaussian Softmax

Model, didn't yield satisfying results.

To generate a sentence that represents a topic one simply feeds the conditional LM a topic vector that is one-hot encoded to the desired topic. The latter thus represents a document that fully consists out of 1 topic. The rest of the process of generating sentences is similar to a normal language model as described in the previous section.

4.5 Experimental set-up

In this chapter we will try to answer the last research question: *Is a language model conditioned on the latent space of a topic model able to generate the most frequently asked questions?* To do this we will first get familiar with how language models work after which we will use the conditioned language model on the same datasets as used for the topic models and look how well the generated sentences relate to the topics from the topic model.

We start by doing experiments with a standard neural language model such that we know that the implemented language model has low perplexity on the test set and is able to generate sentences. After this we will adept this language model such that it can be conditioned on a topic model. We try three different ways to condition language model as described in the previous section. A LM that is conditioned on the input (LM-C-input), a language model conditioned on the output (LM-C-output) and a language model conditioned on a topic model by creating an export for every topic (LM-C-MoE).

For the language model we used the Penn Treebank (PTB), Associated Press News (APNews) and the ABN-AMRO dataset. We also tried 20Newsgroups first but found that the language model was too much affected by artifacts of the data which made it unstable in generating coherent text. We choose PTB because it is the language modeling standard which allows us to compare its perplexity with other language models. We use AP-NEWS because it is a bigger dataset that is more coherent than 20Newsgroups. This dataset is also used by Wang et Al. (2017) to train their conditional language model. Only the latter two models will used for the conditional language model. Using the same datasets for the language models and conditioned language models allows us to compare perplexities and determine whether the added topic information contributes to the predictive performance of the model.

To make the dataset useful we built a dataset where each datapoint had both a BoW representation and a representation where each word was tokenized. We replaced the rare words by and words not present in the trainingset UNK tokens. This to reduce the complexity of the Softmax and to make inference on unseen data possible.

To evaluate the models we look at the predictive performance, the coherency of the sentences it generates and how well the generated sentence relates to the topic it is conditioned on. The predictive performance will be evaluated using perplexity. The last two components will be evaluated qualitatively. In the case of coherency we look at both semantic meaning and grammatical correctness.

Both the standard language model and the conditioned language model are implemented in Tensorflow. The regular language model is based on the code of Zaremba, Sutskever, and Vinyals, 2014 who explores optimal tuning for language models. For the conditional language models no public implementations were available. The models were run on the same hardware as the topic models.

4.6 Results

4.6.1 Language model

To see how a regular language model (LM) performs we started by using the parameters from Zaremba (2014). Although they should be the optimal parameters for PTB, we still wanted to see how it responds to different configurations. In figure 4.7 the performance is depicted in three graphs. **(Note: throughout this chapter we depict test perplexity next to the validation perplexity. The test perplexity is merely depicted to provide information on how this relates to the validation perplexity. Only the validation perplexity is used to determine the best model.)** It shows how the optimal configuration for the validation set is 2 layers with 650 hidden neurons, exactly like the Zaremba (2014) suggests. We see that the model starts overfitting when the number of hidden units exceeds 650. One could choose a configuration with more hidden neurons and achieve better results when adjusting the regularization, however, we chose the medium configuration suggested by Zaremba because of our limited computational resources and time constraints. On

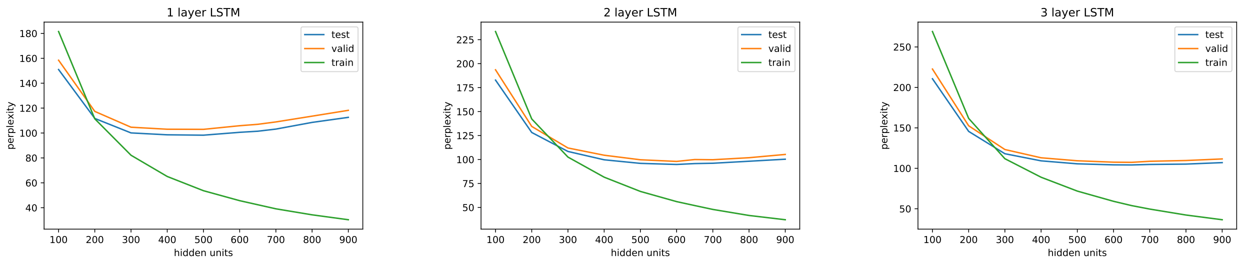


Figure 4.7: LSTM language model results on PTB dataset with different configurations. We observe that the LSTM's are overfitting, especially when the number of hidden units increase.

top of this, achieving lower perplexity doesn't necessarily help us to answer our research question.

When we fix these parameters and change the dropout we also find that 0.5 keep probability is optimal, as the paper suggests. Having a higher keep probability will cause overfitting. This experiment is depicted in figure 4.8.

Since perplexity is not the goal, sampled sentences are depicted for both the PTB and the ABN-AMRO dataset in table 4.1 and table 4.2. In the first table we see a sampled sentence for PTB for every 10 different dropout configurations. Looking at the sampled sentences the perplexity and coherency seem to have some relation. For example, the first sentence with 364 perplexity is a fully incoherent sentence whereas the 5th sentence with 99 perplexity is rather coherent. Figure 4.2 shows a sampled sentence per trained epoch with its corresponding perplexity for the ABN-AMRO dataset. Here we observe how the language model overfits after 4 epochs. What we also see are sentences that look meaningful. Reading closer it seems the language model has learned to be grammatically correct, however, just like the PTB sentences they are meaningless.

Judging by the perplexity and quality of the samples we have a language model that is able to generate samples that resembles the data distribution.

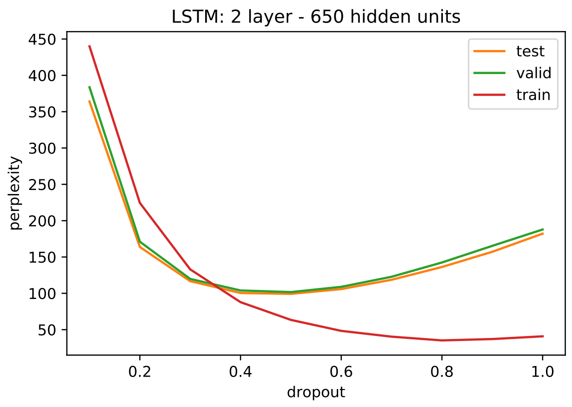


Figure 4.8: LSTM language model results on PTB dataset for different dropout values. We observe that the model achieves the lowest validation and test perplexity with a keep probability of 0.5.

4.6.2 Conditioned Language Models

In this sections we will look at the results of the three conditioned language models. A LM that is conditioned at the input (LM-C-input), a language model conditioned at the output (LM-C-output) and a language model conditioned on a topic model by creating an export for every topic (LM-C-MoE). First we only wanted to look at the LM-C-MoE, but because of the poor results we decided to try different approaches as well.

All language models are tuned like the medium configuration from Zaremba (2014). Although the optimal parameters are probably different for these datasets and models, we didn't have the computational resources and time to do a grid search on all parameters. This means that for all language models we used: 650 hidden units, 2 layers and 0.5 keep probability for the dropout.

dropout	train	valid	test	sampled topic
0.1	440	384	364	yield one salomon with japanese plus our unk books has
0.2	224	171	164	but investors and critics are a pretty proceeding increase to
0.3	134	120	117	a united spokesman said the oas is n't concerned that
0.4	88	104	100	the N N charter raised the problem a unk more
0.5	**63**	**102**	**99**	**all for the first time the official said the company**
0.6	48	109	106	golden nugget has proposed rights of more than $ N
0.7	40	123	119	unk the problem is the prospect problem eos the record
0.8	35	142	136	for instance those who is the unk of those vivid
0.9	37	165	157	though a successor check has been a big journalist attraction
1.0	41	188	182	growth splitting can be a bright problem about being one

Table 4.1: sampled sentences LM with PTB. The numbers in the dropout column represent the keep probability.

epoch	train	valid	test	sampled topic
1	117	85	84	bepalen ontvangen met incassobureau van UNK a betaal van medewerker kan opnemen nog . EOS
2	72	74	72	geadviseerd om met hr te kopen . EOS hier al klant goede deur bij ging op vakantie
3	60	74	71	bm antwoord UNK gekregen (UNK kreeg van mijn pas . EOS nu is het probleem : automaat overboekingen is nog
4	**54**	**73**	**70**	**klant heeft op de richtlijnen maar dus ook gedaan , pas is wel naar de app worden ik partner UNK toe**
5	52	74	71	klant heeft brief van UNK . EOS bc adres is op morgen zijn rekening kan niet afgekeurd gestolen UNK usd om kantoor te regelen .

Table 4.2: sampled sentences from language model trained on the ABN AMRO dataset. We see that the perplexity decreases after which it overfits after 4 epochs.

	LM	LM-C-MoE	LM-C-input	LM-C-output
train	212	591	213	172
valid	172	569	168	139
test	169	566	165	**137**

Table 4.3: Results perplexity for the LM and conditional LM's on the APnews dataset. The conditional language models are conditioned on the topic distribution vector of LDA given the document.

In table 4.3 one finds an overview of the perplexity results that the models achieve on the APNews dataset. The first observation is that the LM-C-MoE has the lowest perplexity performance out of the language models conditioned on the topic distribution vector of lda given the document. It also performs worse than a regular LM which is why we assume there is something wrong with our implementation. The other conditioned language models are able to achieve a lower perplexity than the regular LM which shows that the global information from the topic vector is useful. The model where the topic vector is appended to the output of the LSTM has a perplexity that is significantly lower than the model where the language model is appended to the input. This makes sense because the topic vector remains the same in the entire document. For this reason the LSTM doesn't have to remember anything from the topic vector of a time step earlier which makes the LSTM unnecessary. Since Neural networks are known to have trouble to pass the identity of the input, hence the introduction of Residual and Highway networks, and the allocation of connections to pass the identity of the topic vector, LM-C-input can't match the performance of LM-C-output.

Now we know that the predictive performance is best of the LM-C-output we will have a look at how well they perform qualitatively. In table 4.4 you find sampled sentences from every model trained on APnews conditioned on three topics from LDA. We classify the topics as Israel, geography and the financial market. The LDA simply returns the most likely words given the topic as we've seen in the previous chapter. The sampled sentences of LM-C-MoE seem to produce sentences that are slightly more coherent than random words and on top of they do not resemble the topic in any way. The sampled sentences from LM-C-Input are a lot better from

	topic	sampled topics
LDA	Israel	israel israeli middle occupied east arab minister palestinian bank west
	geography	california los angeles san texas york southern new ohio south
	market	stock wall exchange jones average market shares crash street index
LM-C-MoE	Israel	shultz they among the , , “ to said moslem told should paul bank from be the
	geography	much while of northeastern states the their study , UNK fell the of palestinians held up . , the casino of the entirely
	market	UNK than for production UNK UNK security EOS might where EOS
LM-C-input	Israel	” at give full in the new UNK . EOS “ but free had a result on any . ” EOS “
	geography	” brown practiced withchrysler said . EOS resolutions purchased in the UNK old jet beach , which UNK room study of the liberal had other active compared with a museum it
	market	study on friday to make up miss spence . EOS UNK , apparently the area , not not include the bar
LM-C-output	Israel	EOS shimon shamir has told arafat prime rights talks hafez delegation israel syria west bank gaza arafat yitzhak shamir entered shultz
	geography	oregon parts of southern and 4,500 , western ; scattered and 40s snow in new york , oklahoma
	market	misdemeanor outnumbered losers of investors’mood rules . EOS volume listed market to the american institutions marked

Table 4.4: Sampled sentences from conditional language models conditioned on three topics from LDA: Israel, geography and financial markets

a grammatical perspective, however, they also don’t resemble the topic in any way. The sampled sentences from LM-C-output however, seem to represent the topic as well as be slightly grammatically coherent. For example, the sampled sentence representing the topic Israel contains several words also returned by the topic model, such as Israel and west bank.

One of the common mentioned weaknesses of language models is that they perform poor on out of domain data. The effect is already strong on similar datasets. The perplexity of a language model trained on Dow-Jones newswire doubles when it is applied on the Associated Press newswire from the same period (Roni Rosenfeld, 1996). In our case this isn’t a weakness but something we would like to exploit. The models used so far all used dropout to avoid overfitting on the training set. This is useful when you have a sample of the dataset but want it to represent the ‘full language distribution’. Since we are not interested in building a model that works well on unseen data but in creating sentences that represent this specific dataset it seems that overfitting is not an issue. For this reason we wanted to do an experiment where we remove regularization to optimize the model for this specific dataset.

In table 4.6 we show both the results of the LM-C-output, the best performing model, with and without dropout. We observe that model without regularization outperforms the existing model on both the train perplexity but also for the test perplexity. This seems to suggest that 0.5 is not the optimal dropout for the APNews dataset for this model. What is also interesting is that the perplexity for the regularized model is lower on the test set than the train set. This seems to suggest that the model is not finished training or that the test set data contains more standard language. Both models have been selected by taking the model with the lowest validation perplexity. When the validation error starts to increase the code runs two more epochs to make sure it doesn’t decrease again, this might have been insufficient, even though it had run for 16 hours on a gtx 1070 GPU.

In table 4.6 we compare both the regularized model and unregularized model qualitatively. It seems that the unregularized model more often chooses the UNK token than the regularized one. We found this in this example but also throughout all samples created during training. Why this is the case we do not understand. On top of this it seems to perform similarly by generating somewhat coherent sentences that represent the topic.

Now we know that LM-C-Output is able to generate sentences that somewhat represent a topic we will apply this model to the ABN-AMRO dataset. In table 4.7 the quantitative results for the LM and LM-C-output are depicted. Here we see that the LM-C-output can utilize the extra global features from LDA even though most sentences are short, which results in a lower perplexity. In table 4.8 sampled sentences for four of the topics of the ABN-AMRO data set are depicted. The sentences seem less coherent than earlier samples. This is probably

	LM-C-output: keep=0.5	LM-C-output: keep=1.0
train	172	69
valid	139	101
test	137	**99**

Table 4.5: Perplexity on the APNews dataset. This experiment explores the impact of regularization. *keep* stands for the keep probability for dropout.

	topic	sampled topics
LDA	Israel	israel israeli middle occupied east arab minister palestinian bank west
	geography	california los angeles san texas york southern new ohio south
	market	stock wall exchange jones average market shares crash street index
LM-C-output keep=0.5	Israel	EOS shimon shamir has told arafat prime rights talks hafez delegation israel syria west bank gaza arafat yitzhak shamir entered shultz
	geography	oregon parts of southern and 4,500 , western ; scattered and 40s snow in new york , oklahoma
	market	misdemeanor outnumbered losers of investors'mood rules . EOS volume listed market to the american institutions marked
LM-C-output keep=1.0	Israel	: israel radio occupied arab territories occupied territories in thewest UNK israel 's jewish middle east peace corps UNK lebanon palestinians
	geography	jesse jackson suffered only UNK inches of snow showers wednesday morning at los angeles day of new york , while east
	market	as pause thursday showed up to UNK hours in UNK of UNK financier alan stephens

Table 4.6: sampled sentences from LM-C-output trained on APnews. This experiment explores the impact of regularization. *keep* stands for the keep probability for dropout.

because we used a sample of 10,000 documents, and since they are shorter than the APNews documents the LM has less data to work with. However, also here the model is able to capture the semantics of the topic. In topic 11 this is most apparent because this topic represents the English questions, and the sampled sentence from this topic is also entirely in English. Also in the other topic 4 it is clear that the sampled sentence and the topic are related. We see this because of the overlapping words in the lda topic and the generated sentence from that topic.

Even though the topic model is able to find the latent topics in the data and the LM-C-Output shows that it is able to generate sentences that represent those latent topics. When analyzing the generated results from the LM-C-output on the ABN AMRO dataset, we don't get the feeling that we found the most frequently generated sentences in the dataset. Does this mean that the model is unsuitable for finding the most frequently generated sentences? Not necessarily, besides the fact that the topic model finds a few uninterpretable topics and the generated sentences could potentially be from better quality. We think that the poor results are partly caused by the amount of noise that is present in the dataset. If the data would consist from only the posed sentences without the noise of further added information about the conversation, we think that the topics and generated sentences would be more coherent and maybe even useful, which is not the case for these results.

	LM	LM-C-output
train	75	55
valid	66	62
test	66	**59**

Table 4.7: Perplexity for both the standard LM and LM-C-Output on the ABN-AMRO dataset. Here we use the same parameters for the LM and the C-LM-output except that the latter has no dropout.

	topic	sampled topics
LDA	1	basis pasje volmacht defect algemeen inkomen adreswijziging uitzoeken geregeld krijgt
	4	kl direct rek omgezet saldo sparen dient overgemaakt aanwezig rabo
	11	abn amro to the bank on for and card please
	18	number klant rekening graag meneer bedrag kosten weten rente hypotheek
LM-C-output	1	van pl van UNK wil dit in het activeren dag worden geweest door mw in pas ? EOS
	4	rek rek saldo saldo saldo rek saldo omgezet saldo saldo kl rek omgezet saldo saldo binnen direct sparen
	11	to an it . EOS UNK UNK our UNK . EOS i have maken dear have a from UNK a UNK that ? EOS
	18	van meneer was op NUMBER , UNK % UNK mogelijk . EOS UNK zal dat het de UNK niet duidelijk . EOS

Table 4.8: sampled sentences from LM-C-output trained on the ABN-AMRO dataset.

4.6.3 Exploration Latent Space

In this section we will have a closer look at the latent topic space of the documents. In the previous section we explored the boundaries of the space by generating sentences conditioned on a single topic. In this section we will look how the documents are distributed over the space and whether the space within the boundaries is also meaningful for our language model.

To visualize the multidimensional space of the topic distributions we use t-Distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and G. Hinton, 2008). This method allows us to reduce a k-dimensional space into a lower dimensional space such that we can understand how the data is distributed.

In figure 4.9 the latent topic space of a sample of documents from APNews is visualized using t-SNE. The topic space is computed using a trained LDA topic model with 20 dimensions. We see that the data is relatively clustered, which means that the documents are separable within the topic space.

By clicking on the dots that represent the documents, the LM-C-output model generates a piece of text conditioned on the topic distribution of the document. We see that on the bottom right corner the LM generates sentences that both start with the word ‘a’ and uses the words ‘convention’. We observe similar behavior in the rest of the space. This shows that the language model did not overfit on certain topic distributions but is able to traverse the latent topic space and generate sentences regardless on whether the model has seen such a topic distribution before.

Another thing we observe is that the sentences that are close to one another are also semantically similar. Take the top right cluster for example, the sentences are different, but both are related to the domain of politics.

Besides generating from topic distributions from documents we can also sample throughout the space. In table reftable:homotopy we generated sentences from the latent space between two topics. As with all sentences produced by the model, the quality of the sentences in between is rather poor. Nevertheless, in this experiment we observe that the semantic content gradually changes. The latter is mostly apparent in the sentence closest to the economy topic.

From these observations we can conclude that the language model works as desired as well as that the topic model is able to create a smooth latent topic space.

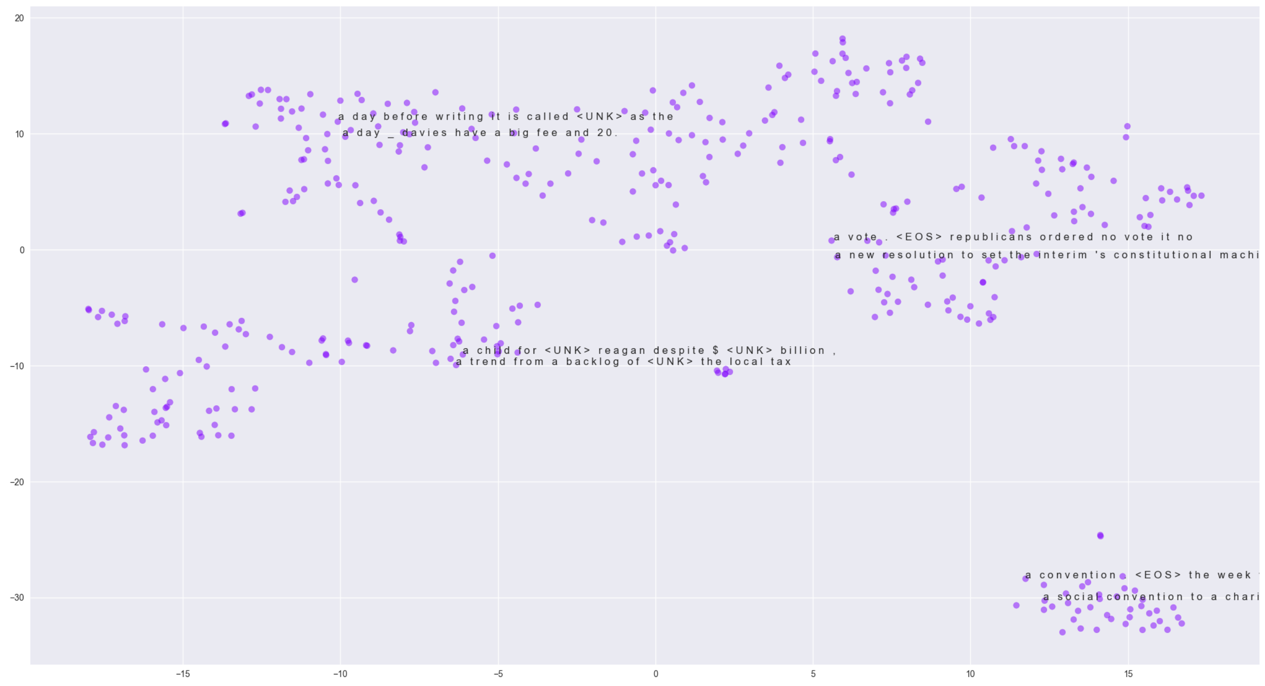


Figure 4.9: t-SNE over LDA topic distributions of sampled APnews documents. The sentences are generated by the LM-C-output model conditioned on the topic distributions.

	sampled sentences
Economy	**a half of the economy but UNK p.m. on the**
	a market in UNK , from key points on the
	a lengthy loss of the new construction UNK in the
	a shipment perhaps by the search of more open the
	a position UNK 's order to rescue credit corp.
	a alleging . EOS " i was very afraid that
Israel	**a permanent jewish palestinian israel against court organization secrets between**

Table 4.9: Generated sentences using LM-C-output between two topics. The intermediate sentences represent a linear interpolation between the two topics. We observe that the semantic content shifts gradually from the economy to Israel.

4.7 Conclusion & Discussion

In this chapter we explored three ways to condition a language model on the topic vector of a topic model to answer the last research question: *Is a language model conditioned on the latent space of a topic model able to generate the most frequently asked questions?*. We found that only the LSTM-C-output model, the model that added the topic distribution before the output layer, was successful. This model was able to use the global feature from the topic model to improve its predictive performance on both the APNews and ABN AMRO dataset. The model was also able to generate sentences that represented the topics found by lda. However, the sentences were still clearly generated by a machine and semantically incomprehensible.

These don't seem to be useful features for a system that should generates the most frequently asked questions. We think that the data quality is one of the reasons that the output is of poor quality. It would be interesting to see what would happen if the model is trained on a dataset with only the questions without any other information. Until this is done we can't give a conclusive answer to the research question. However, we can say that it shows potential given that the output represents the topics found. We can also say that with suboptimal training data it is certainly not able to generate the most frequently asked questions. For now this method don't seem to be the best way for the bank to find the most frequently asked questions and better results might even be achieved by simply sampling sentences from documents that score high for the given topic.

The model could probably still be improved by better tuning the model. Now we used the recommended parameters for the PTB dataset which are not the optimal parameters for the used datasets. In our case it was not possible because of time constraints and computational resources to do a proper grid search over all possible parameters. However, we think our conclusions to the research questions would not be different with a optimally tuned model. A way that would probably help to find better quality sentences is using beam search instead of greedy search to find the best sentences.

Chapter 5

Discussion & Conclusion

In this thesis we set out to automate the process of finding the most frequently asked questions in a dataset. To achieve this we put down four research questions which should give us clarity on whether this is possible and if so, what would be the optimal way to do so with the existing ML toolbox. In this chapter we provide an overview of the findings related to the research questions and hypothesis. We will also discuss the limitations of our research and possible future research.

5.1 Research Questions and Hypothesis

In this section we provide an overview of the answers to our research questions and reflect on our hypothesis.

RQ 1: How to find the most frequently asked questions from a large set of questions?
We answered this by exploring the entire option space within the NLP domain. We concluded that a combination of a topic model with a language model seems most promising given the data made available by the ABN AMRO Bank. The topic model finds the types of questions and the language model generates sentences that represent the most frequently asked questions given the topic.

RQ 2: Is a topic model a suitable choice to distinguish different kind of questions?
We explored this question by exploring how topic models work as well as implementing and experimenting with several of them. We conclude that a topic model is a suitable choice to distinguish different types of questions as long as it is not a problem if a number of the topics have no interpretable meaning.

RQ 3: What topic model is the most suitable choice to distinguish different types of questions?
For this question we experimented with LDA, NVDM and Prodlda and concluded that LDA is the most suited topic model because it's robust and has the highest topic coherence. We also observed that Prodlda achieved results close to that of LDA and was therefore the most promising neural topic model.

Because we observe that the dataset from ABN AMRO contains mostly short documents, and questions are short in general, we adjusted the Prodlda model by replacing the bag of words encoder with an LSTM encoder. For this we tried three different versions. From these versions the bi-LSTM encoder that uses the mean of the outputs yields the best performance in combination with Prodlda. We showed that this model, Prodlda with a bi-LSTM encoder, can both outperform LDA and Prodlda for topic coherence on datasets with short documents (<30 words).

RQ 4: Is a language model conditioned on the latent space of a topic model able to generate the most frequently asked questions?
To answer the last question we tried three different ways to conditioning a language model on a topic distribution. We tried a model which uses a mixture of experts approach to condition on the topic distribution, a model that conditioned at the input and a model that conditions at the output. Only the model conditioned at the output was able to both generate grammatically viable sentences and generate sentences that semantically represent the topics. We also found that using a LM conditioned on a topic model was able to achieve a lower perplexity than without.

Although the model is capable of generating sentences that resemble the topics, using the model on the ABN AMRO dataset didn't yield a set of the most frequently asked questions. One of the main reasons for this is probably the quality of the data. The data didn't only contain the questions but also other descriptions of the conversation which both leads to topics that don't resemble types of questions as well as generated text that is not a question but rather a description.

Hypothesis: A combination of a topic model with a language model is an effective option to find the most frequently asked questions.
On the one hand our model our model didn't find an easily interpretable list of the most frequently asked questions. For this reason one could say that we should reject the hypothesis. On the other hand we know that the data we had was far from ideal. To generate the most frequently asked questions we probably need a dataset that consists only out of questions. Additional research is necessary to see if our LM-C-output model would find the frequently asked questions if such a dataset is available. Therefore we neither accept or reject the hypothesis.

It should be noted that this method - even with ideal data - will most likely not be able to generate a list of frequently asked questions that one can put on the website. However, the experiments show that the semantic information of the topic is captured in the language model. This indicates that with better data we can find valuable insights using this model.

5.2 Discussion

In this section we will discuss the limitations of our research, our approach and recommendations for others.

Tuning & Data Size
Because most of our used models are computationally expensive to run we were not able to do a proper grid search over the parameters to have an optimally tuned model. Even though we used the parameters from the authors, we observed that they were not the optimal parameters for our datasets. For the same reason we only apply our model on a sample of the data. Since neural topic models benefit from larger datasets this could have had a negative impact on the results of our neural models. These factors can be the reasons why Prodlda for example, has a lower topic coherence than LDA in our findings.

Sentence Generation
The quality of our generated sentences could have also been better if we used beam search to find the optimal sentence instead of greedily creating the sentences. We tried to use the implementation by Tensorflow, but this turned to be hard to integrate in our implementation. This is definitely something one wants to use in a production system for optimal results.

Topic Model
As earlier mentioned the topic models find topics that are not interpretable by humans which is a disadvantage of topic models. Another disadvantage is that they have a fixed number of topics which forces the model the join topics in an unnatural way. There exist methods to find the optimal number of topic models based on perplexity. While perplexity perhaps not be the metric one wants to optimize, this could a better approach than arbitrarily picking the number of topics.

Applied Approach
The thesis is mainly focused on applying existing methods. This was a conscious decision for several reasons. Firstly, the chances of creating a model that performs better and gets adopted by the community is very tiny given the sheer number of publications each year and the limited experience of the author. Secondly, because the goal to create models is to use them. Most research is dedicated on public datasets, which is useful because we can compare the findings. However, when people try to use them on different datasets they can behave different than they expected. Although the UvA is not a university of applied science, the models used are still too complex to be used at the applied universities. For this reason the Dutch universities have a responsibility in doing applied research such that companies, besides the tech giants, can leverage the power of Deep Learning.

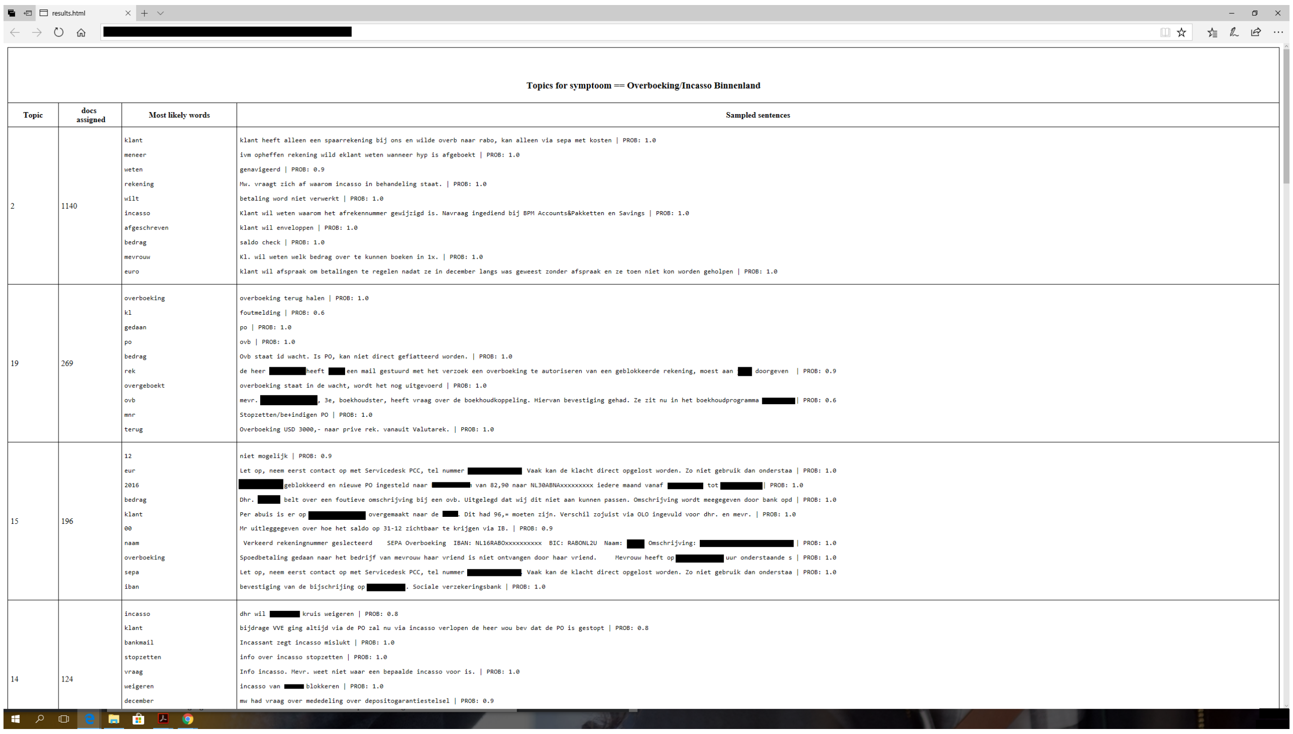
Data Quality
As mentioned in the conclusion the quality of our data is identified as one of the possible reasons for the poor performance of the model. Most research currently focuses to incrementally improving existing models instead of creating valuable datasets. Fei-Fei Li, the creator of the Imagenet dataset, was first ridiculed at Stanford for allocating her time to create a dataset. Now we realize Imagenet has been one of the key drivers behind the deep learning revolution. This research once again proofs how important it is to have the right dataset and underlines that more effort should go to create better datasets.

General Data Protection Regulation

The enactment of the General Data Protection Regulation can also not go unmentioned. The data we dealt with contained personal information. This is why we handled it with utmost care and deleted our data after using it. For our small research this was not an obstacle, however, if one wants to keep such a dataset it is not possible anymore. This because when a customer wants to be deleted, all information of this person should be deleted. There is no way that this can be done when data scientists throughout the company all have their own excerpts from the databases which continue to exist on their local work station. For this reason reliable ways should be found to remove personal information from textual datasets such that people can save these datasets for longer periods.

The Simpler Way
A common joke is that one should use the words Artificial Intelligence while fund raising, Machine learning when hiring and linear regression in production. Even though some companies, especially larger ones, start to use more complex models it is still true that the simplest models can be the most effective ones. In my first month I took out a day to build a small tool to create some insight in the dataset. This tool, visualized in figure 5.1, clusters the ABN AMRO dataset using LDA and visualizes these topics on a website. By assigning documents to their largest topics we can order the topics for importance. On top of this it shows the most probable words given the topic and the first 30 words of the documents that have the highest probability given the topic. Although the tool was simple, in retrospect, it seems more useful than the model which took multiple months to build.

We should realize that deep learning is still in its infancy and very hard to get to a production environment. For most companies, the easiest solution is still the best solution. Both fundamental and applied research is necessary to deliver on the promises the AI community has made throughout recent years, and we hope that this thesis can contribute to bridge the gap between the promise and reality.



Topics for symptoom == Overboeking/Incasso Binnenland

Topic	docs assigned	Most likely words	Sampled sentences
2	1140	klant	klant heeft alleen een spaarrekening bij ons en wilde overb naar rabo, kan alleen via sepa met kosten | PROB: 1.0
		meneer	ivm opheffen rekening wild eklant weten wanneer hyp is afgeboekt | PROB: 1.0
		weten	genavigeerd | PROB: 0.9
		rekening	Mw. vraagt zich af waarom incasso in behandeling staat. | PROB: 1.0
		wilt	betaling word niet verwerkt | PROB: 1.0
		incasso	Klant wil weten waarom het afrekennummer gewijzigd is. Navraag ingediend bij BPM Accounts&Pakketten en Savings | PROB: 1.0
		afgeschreven	klant wil enveloppen | PROB: 1.0
		bedrag	saldo check | PROB: 1.0
		mevrouw	Kl. wil weten welk bedrag over te kunnen boeken in 1x. | PROB: 1.0
		euro	klant wil afspraak om betalingen te regelen nadat ze in december langs was geweest zonder afspraak en ze toen niet kon worden geholpen | PROB: 1.0
19	269	overboeking	overboeking terug halen | PROB: 1.0
		kl	foutmelding | PROB: 0.6
		gedaan	po | PROB: 1.0
		po	ovb | PROB: 1.0
		bedrag	Ovb staat id wacht. Is PO, kan niet direct gefiatteerd worden. | PROB: 1.0
		rek	de heer [redacted]heeft [redacted] een mail gestuurd met het verzoek een overboeking te autoriseren van een geblokkeerde rekening, moest aan [redacted] doorgeven | PROB: 0.9
		overgeboekt	overboeking staat in de wacht, wordt het nog uitgevoerd | PROB: 1.0
		ovb	mevr. [redacted], 3e, boekhoudster, heeft vraag over de boekhoudkoppeling. Hiervan bevestiging gehad. Ze zit nu in het boekhoudprogramma [redacted]| PROB: 0.6
		mnr	Stopzetten/be+indigen PO | PROB: 1.0
		terug	Overboeking USD 3000,- naar prive rek. vanauit Valutarek. | PROB: 1.0
15	196	12	niet mogelijk | PROB: 0.9
		eur	Let op, neem eerst contact op met Servicedesk PCC, tel nummer [redacted] Vaak kan de klacht direct opgelost worden. Zo niet gebruik dan onderstaa | PROB: 1.0
		2016	[redacted]geblokkeerd en nieuwe PO ingesteld naar [redacted] van 82,90 naar NL30ABNAxxxxxxxxx iedere maand vanaf [redacted] tot [redacted]| PROB: 1.0
		bedrag	Dhr. [redacted] belt over een foutieve omschrijving bij een ovb. Uitgelegd dat wij dit niet aan kunnen passen. Omschrijving wordt meegegeven door bank opd | PROB: 1.0
		klant	Per abuis is er op [redacted] overgemaakt naar de [redacted]. Dit had 96,= moeten zijn. Verschil zojuist via OLO ingevuld voor dhr. en mevr. | PROB: 1.0
		00	Mr uitleggegeven over hoe het saldo op 31-12 zichtbaar te krijgen via IB. | PROB: 0.9
		naam	Verkeerd rekeningnummer geslecteerd SEPA Overboeking IBAN: NL16RABOxxxxxxxxxx BIC: RABONL2U Naam: [redacted] Omschrijving: [redacted]| PROB: 1.0
		overboeking	Spoedbetaling gedaan naar het bedrijf van mevrouw haar vriend is niet ontvangen door haar vriend. Mevrouw heeft op [redacted] uur onderstaande s | PROB: 1.0
		sepa	Let op, neem eerst contact op met Servicedesk PCC, tel nummer [redacted]. Vaak kan de klacht direct opgelost worden. Zo niet gebruik dan onderstaa | PROB: 1.0
		iban	bevestiging van de bijschrijing op [redacted]. Sociale verzekeringsbank | PROB: 1.0
14	124	incasso	dhr wil [redacted] kruis weigeren | PROB: 0.8
		klant	bijdrage VVE ging altijd via de PO zal nu via incasso verlopen de heer wou bev dat de PO is gestopt | PROB: 0.8
		bankmail	Incassant zegt incasso mislukt | PROB: 1.0
		stopzetten	info over incasso stopzetten | PROB: 1.0
		vraag	Info incasso. Mevr. weet niet waar een bepaalde incasso voor is. | PROB: 1.0
		weigeren	incasso van [redacted] blokkeren | PROB: 1.0
		december	mw had vraag over mededeling over depositogarantiestelsel | PROB: 0.9

Figure 5.1: HTML output of LDA on ABN AMRO data. It gives further insight in the topics by assigning the documents to the topic by taking the max of the topic vector. It also shows the topics and the start of the documents that consists most of this topic. The topics are ordered to the number of topics assigned to them to provide fast insight. (personal information has been removed)

5.3 Future Research

In this section we suggest research directions that we think could lead to better results.

Same Models with More Specific Data and Tuning
As mentioned in the discussion, both the data and the tuning were not ideal. By acquiring a more specific

dataset of questions and doing a search over all parameters of both the topic and the language model, the model is most likely to generate more coherent topics as well as more fluent generated sentences.

Condition Topic Model on Author
One of the things we observed in the data is that all call-center operators have their own style of efficiently describing the conversation they had. These different styles make it difficult for the topic model to only distinguish semantic content and we observe that it also fits topics on styles of the authors. For this reason we think it is interesting to make the topics independent of the authors by conditioning the topic model on the authors as well. We propose to do this by concatenating a vector that represents the author to the input and the output of the neural topic model.

Convolutional Encoder for Prodlda
In this research we looked at LSTM encoders for Prodlda to leverage the local information of the document. It would be interesting to see how this compares to a convolutional encoder because the sliding window approach is more similar to the traditional n-gram models.

Clustering Documents in Latent Space
At this moment we generate sentences that represent the individual topics found by the topic model. What we could also do is cluster the documents based on their topic distribution and generate sentences conditioned on the vectors that represent the means of those clusters. By doing so we can count the number of documents belonging to a cluster which is not possible using a topic model. How this method will behave we don't know since it's basically a clustering over a clustering.

Bibliography

Adji B. Dieng Chong Wang, Jianfeng Gao and John William Paisley (2016). “TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency”. In:

Arora, Sanjeev et al. (2013). “A practical algorithm for topic modeling with provable guarantees”. In: *International Conference on Machine Learning*, pp. 280–288.

Baeza-Yates, Ricardo, Berthier Ribeiro-Neto, et al. (1999). *Modern information retrieval*. Vol. 463. ACM press New York.

Bahl, Lalit R et al. (1990). “A tree-based statistical language model for natural language speech recognition”. In: *Readings in Speech Recognition*. Elsevier, pp. 507–514.

Bengio, Yoshua et al. (2003). “A neural probabilistic language model”. In: *Journal of machine learning research* 3.Feb, pp. 1137–1155.

Bird Steven, Edward Loper and Ewan Klein (2009). “Natural Language Processing with Python”. In:

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe (2017). “Variational inference: A review for statisticians”. In: *Journal of the American Statistical Association* 112.518, pp. 859–877.

Blei, David M and John D Lafferty (2009). “Topic models”. In: *Text Mining*. Chapman and Hall/CRC, pp. 101–124.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). “Latent Dirichlet Allocation”. In: *J. Mach. Learn. Res.* 3, pp. 993–1022. ISSN: 1532-4435. URL: `http://dl.acm.org/citation.cfm?id=944919.944937`.

Bowman, Samuel R et al. (2015). “Generating sentences from a continuous space”. In: *arXiv preprint arXiv:1511.06349*.

Cao, et Al. (2015). “Ranking with Recursive Neural Networks and Its application to multi-Document Summarization”. In:

Chang, Jonathan et al. (2009). “Reading tea leaves: How humans interpret topic models”. In: *Advances in neural information processing systems*, pp. 288–296.

Chelba, Ciprian et al. (2013). “One billion word benchmark for measuring progress in statistical language modeling”. In: *arXiv preprint arXiv:1312.3005*.

Della Pietra, Stephen et al. (1992). “Adaptive language modeling using minimum discriminant estimation”. In: *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on.* Vol. 1. IEEE, pp. 633–636.

Dhino Hardaya, surjandari and Hardono. “clustering and visualization of community complaints and proposals using text mining and geographic information system”. In:

Erkan, Günes and Dragomir R Radev (2004). “Lexrank: Graph-based lexical centrality as salience in text summarization”. In: *Journal of Artificial Intelligence Research* 22, pp. 457–479.

Glorot, Xavier and Yoshua Bengio (2010). “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256.

Griffiths, Thomas L and Mark Steyvers (2004). “Finding scientific topics”. In: *Proceedings of the National academy of Sciences* 101.suppl 1, pp. 5228–5235.

Gubbins, Joseph and Andreas Vlachos (2013). “Dependency Language Models for Sentence Completion.” In: *EMNLP*. ACL, pp. 1405–1410.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Hofmann, Thomas (1999). “Probabilistic latent semantic analysis”. In: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 289–296.

James, Frankie (2000). “Modified kneser-ney smoothing of n-gram models”. In: *Research Institute for Advanced Computer Science, Tech. Rep. 00.07*.

Jozefowicz, Rafal et al. (2016). “Exploring the limits of language modeling”. In: *arXiv preprint arXiv:1602.02410*.

Kernighan, Mark D., Kenneth W. Church, and William A. Gale (1990). “A Spelling Correction Program Based on a Noisy Channel Model”. In: *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*. COLING ’90. Helsinki, Finland: Association for Computational Linguistics, pp. 205–210.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kingma, Diederik P. and Max Welling (2013). "Auto-Encoding Variational Bayes." In: *CoRR* abs/1312.6114. URL: `http://dblp.uni-trier.de/db/journals/corr/corr1312.html#KingmaW13`.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov, pp. 2579–2605.

Martín Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* Software available from tensorflow.org. URL: `https://www.tensorflow.org/`.

Miao, Yishu, Edward Grefenstette, and Phil Blunsom (2017). "Discovering discrete latent topics with neural variational inference". In: *arXiv preprint arXiv:1706.00359*.

Miao, Yishu, Lei Yu, and Phil Blunsom (2016). "Neural variational inference for text processing". In: *International Conference on Machine Learning*, pp. 1727–1736.

Mikolov, Tomáš et al. (2010). "Recurrent neural network based language model". In: *Eleventh Annual Conference of the International Speech Communication Association.*

Minka, Thomas and John Lafferty (2002). "Expectation-propagation for the generative aspect model". In: *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc., pp. 352–359.

Nallapati, Ramesh, Feifei Zhai, and Bowen Zhou (2017a). "SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents." In: *AAAI*, pp. 3075–3081.

— (2017b). "SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* Pp. 3075–3081. URL: `http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14636`.

Ng, Andrew Y and Michael I Jordan (2002). "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes". In: *Advances in neural information processing systems*, pp. 841–848.

Paulus, Romain, Caiming Xiong, and Richard Socher (2017). "A deep reinforced model for abstractive summarization". In: *arXiv preprint arXiv:1705.04304*.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Plotz, Thomas and Gernot A. Fink (2009). "Markov Models for Offline Handwriting Recognition: A Survey". In: *Int. J. Doc. Anal. Recognit.* 12.4, pp. 269–298. ISSN: 1433-2833. DOI: `10.1007/s10032-009-0098-4`. URL: `http://dx.doi.org/10.1007/s10032-009-0098-4`.

Porteous, Ian et al. (2008). "Fast collapsed gibbs sampling for latent dirichlet allocation". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, pp. 569–577.

Rios, Miguel, Wilker Aziz, and Khalil Sima'an (2018). "Deep Generative Model for Joint Alignment and Word Representation". In: *arXiv preprint arXiv:1802.05883*.

Rosenfeld, R. (1996). "Two decades of statistical language modeling: where do we go from here?" In: *Proceedings of the IEEE* 88.8, pp. 1270–1278.

Rosenfeld, Roni (1996). "A maximum entropy approach to adaptive statistical language modeling". In:

Sartre. (1946). *Existentialism is a Humanism:* Editions Nagel.

Schwenk, Holger, Anthony Rousseau, and Mohammed Attik (2012). "Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation". In: *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. WLM '12. Montreal, Canada: Association for Computational Linguistics, pp. 11–19. URL: `http://dl.acm.org/citation.cfm?id=2390940.2390942`.

Srivastava, Akash and Charles Sutton (2017). "Autoencoding variational inference for topic models". In: *arXiv preprint arXiv:1703.01488*.

Srivastava, Nitish et al. (2014). "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Teh, Yee W, David Newman, and Max Welling (2007). "A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation". In: *Advances in neural information processing systems*, pp. 1353–1360.

Van Rossum, G. (1995). "Python tutorial, Technical Report CS-R9526". In: *Centrum voor Wiskunde en Informatica (CWI).*

Velzen, Joost van (2017). *Giphart schrijft samen met een robot.* Trouw (newspaper).

Wallach, Hanna M, David M Mimno, and Andrew McCallum (2009). "Rethinking LDA: Why priors matter". In: *Advances in neural information processing systems*, pp. 1973–1981.

Wang, Wenlin et al. (2017). "Topic Compositional Neural Language Model". In: *CoRR* abs/1712.09783. arXiv: `1712.09783`. URL: `http://arxiv.org/abs/1712.09783`.

Warsito and Sugiono (2016). “Determining Citizen Complaints to The Appropriate Government Departments using KNN Algorithm”. In: *ICSITech*.

Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals (2014). “Recurrent neural network regularization”. In: *arXiv preprint arXiv:1409.2329*.

Zhang, LJ et al. (2016). “Keyword extraction algorithm based on improved text rank”. In: *Journal of Beijing Institute of Graphic Communication* 24.4, pp. 51–55.

Zhang, Xiang, Junbo Zhao, and Yann LeCun (2015). “Character-level convolutional networks for text classification”. In: *Advances in neural information processing systems*, pp. 649–657.