Probabilistic Topic Models in Natural Language Processing

Bachelor's Thesis submitted

to

Prof. Dr. Wolfgang K. Härdle

and

Prof. Dr. Cathy Y. Chen

Humboldt-Universität zu Berlin
School of Business and Economics
Ladislaus von Bortkiewicz Chair of Statistics

by

Jérôme Bau
(557833)

in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Economics
Paris, November 22, 2016

Abstract

In machine learning, topic models serve to discover abstract structures in large document collections. I present a tailored selection of concepts both from information theory and from statistics to build a solid foundation for understanding topic models. The concepts presented include theorems as well as examples and visualizations. I focus on two models in particular: The *Latent Dirichlet Allocation* and the *Dynamic Topic Model*. Applications, built in the Python programming language, demonstrate possible cases of application, such as matching news articles similar in content and exploring the topic evolution of news articles over time. This paper's objective is to guide the reader from a casual understanding of basic statistical concepts, such as those typically acquired in undergraduate studies, to an understanding of *topic models*.

Keywords: Topic models, text mining, Latent Dirichlet Allocation, Dynamic Topic Model, Vector Space Model

Contents

List of Figures

1 Introduction

Modern machine learning algorithms in natural language processing often base on a statistical foundation and make use of inference methods, such as *Markov Chain Monte Carlo*, or benefit from multivariate probability distributions used in a Bayesian context, such as the *Dirichlet distribution.* Naturally, understanding the statistical theory behind these powerful models is a must for a proper usage, for understanding, and the avoidance of potential errors and risks of application.

While engineers of different fields have been dreaming of automated knowledge discovery for several decades, only the technological progress of the last decades has allowed for major advances, leading to its application in a vast variety of fields, such as bioinformatics, linguistics and medicine. In the year 1945, the American engineer Vannevar Bush wrote his famous essay "As We May Think", in which he called for "the massive task of making more accessible our bewildering store of knowledge", proposing the idea of a machine that collects text materials and indexes them in a more accessible way, envisioning a knowledge collection that is indexed through trails of association. Since Bush's essay, data sets have grown, computers have become powerful but in principle the goal remained to structure and index large sets of data and to model associations between given documents.

Topic models are one solution to understanding large sets of documents and its relations, which are at the center of this work. I focus on two models in particular, the *Latent Dirichlet Allocation* and its extension *Dynamic Topic Models.*
Understanding these models is understanding the concepts that intertwine with it: Document preprocessing, inference methods, distance measures and probability distributions. This work aims at creating a theoretical foundation for understanding what happens in these particular models and how large document collections are used to train the models - how are parameters inferred? How is document similarity determined? How can we use the models for unseen data?
While detailed literature is available for most of the concepts mentioned in this paper, ranging from document preprocessing to inference methods, I want to contribute to the understanding of topic models by presenting a work that is custom tailored to topic models in the sense that it covers all concepts that I see as essential to understanding topic models, in particular the *Latent Dirichlet Allocation.*

On the basis of this theoretical foundation, I present some examples of applications that I developed in Python. Taking advantage of the easily accessible, massive document collection

Wikipedia, I train a LDA model and use it to understand query documents, such as the US Constitution and to match news articles. Processing German news articles from early 2016, grouped by publishing month, I present how topics evolve over time.

The remainder of this paper is organized as follows. In Section 2, I outline concepts of classical information retrieval that will be of importance to understanding topic models. Section 3 presents concepts of statistics and machine learning that reflect the foundation of the topic models presented in Section 4. In Section 5, I present applications of topic models that I developed in Python. Section 6 contains a conclusion with an outlook on recent use cases of topic models in various fields.

2 Concepts of Information Retrieval

Probabilistic Topic Models are methods of Information Retrieval. This field introduces its own set of terms and concepts, not necessarily linked to the statistical concepts used.
This section gives an overview of text-related key concepts needed to explore Probabilistic Topic Models.

2.1 Basic Terms of text mining and Preprocessing

I introduce the following basic definitions, commonly used in information retrieval. While these definitions might seem trivial, understanding their precise meaning is necessary to explore more complex concepts in this field.

Definition 2.1 (Corpus)**.**
A (large) set of documents.
(similar in Baeza-Yates and Ribeiro-Neto (1999))

Definition 2.2 (Vocabulary)**.**
Let v_i be a distinct index term. Let I be the number of distinct index terms, found in the corpus. The vocabulary, denoted V, of the corpus is defined as $V = \{v_1, ..., v_I\}$. I is its size.
(similar in Baeza-Yates and Ribeiro-Neto (1999))

Definition 2.3 (Query document)**.**
A document, generally not derived from the corpus upon which the model was trained, which is used to receive a model-based response, such as a classification or similarity measure.

Definition 2.4 (Bag-of-words model)**.**
A simplified document representation in which a document is treated as a multiset of its words, i.e. the order of words is not taken into account, while number of word occurrences remains known. (similar in Baeza-Yates and Ribeiro-Neto (1999))

All topic models in natural language processing whose objective is to explore document similarities or latent structures, being probabilistic or algebraic, will begin with *preprocessing* of the given corpus.
The following subsections are derived from a very common preprocessing structure found in many applications and theoretical sources (such as Baeza-Yates and Ribeiro-Neto (1999, Chapter 6)). Some aspects of preprocessing may be added or left out, depending on the objectives of the superjacent model.

2.1.1 Lexical Analysis / Tokenization

We start by seeing a corpus as a collection of strings of varying lengths, where a priori any two characters are of equal value, including word dividers (space in the Latin alphabet) and other punctuation marks.

Tokenization intends to separate the given string into individual words, referred to as *tokens*, which are meaningful text units. While this idea is rather simple, natural language processing models are frequently challenged in recurring exceptions, such as ones caused by hyphenation. Looking at ”San Francisco-based co–worker”, the human eye will easily break apart the string into meaningful segments, such as the tokenization {”San Francisco”, ”based”, ”co–worker”}. The preprocessing algorithm, however, will not easily distinguish the two cases of breaking apart hyphenated text units or not. Numbers and punctuation can equally pose problems. Most preprocessing algorithms will convert all letters to lowercase and treat punctuation and spacing as token dividers. An in-depth analysis of problematic semantics is presented by Fox (1992).

2.1.2 Stop words

After tokenization, we can easily transform the corpus into more convenient formats, such as vectors or matrices containing the tokens (presented in Section 2.2). In regards to topic models, especially on very large corpora, we have a strong interest in reducing the dimension of vocabulary (see Section 2.2), i.e. distinct index terms v_i, to reduce computation complexity. For dimension reduction we take into account that not all words possess equal information content. While intending to reduce the dimension of token vectors or the equivalent matrices, we would like to minimize the loss of information. In consequence, we eliminate words that have little to no meaning, which are referred to as *stop words*. Many sources provide lists of stop words, such as ”the”, ”a” and ”of”. For more details on stop lists, including example lists, see Fox (1992).

Such a stop word list cannot be generalized (e.g. using the 100 most common words), since some cases of application might call for inclusion of case- or source-related stop words.

2.1.3 Stemming

When trying to computationally understand a text corpus, certain variations of a word will contain the same information content. Reducing a token to its *stem* – removing prefixes, suffixes, etc – can increase the model performance while reducing the vocabulary size. This procedure is called *stemming*.

Example: $\{$'expects', 'expected', 'expectation'$\}$ $\longrightarrow$ 'expect'.

The concept of stemming has to be used with caution, since it is highly dependent on the type of application of the model as well as the input language, as stemming is inapplicable to certain languages (Baeza-Yates and Ribeiro-Neto, 1999).

2.2 Vector Space Model

The Vector Space Model is a fundamental concept used in information retrieval since the 1970's and commonly accredited to Gerard A. Salton, a leading researcher in the field of information retrieval at the time.

In general, a vector space model is based on a space with the number of dimensions equal to the size of vocabulary I, so that each document can be represented as an I-dimensional vector.

At the end of this section, I present a visual example.

2.2.1 TF-IDF

The number of appearances of a term in a document is a significant characteristic of the document and is a basis for understanding its nature in many information retrieval models (Baeza-Yates and Ribeiro-Neto, 1999).

In the year 1957, Hans Peter Luhn published a framework for this notion, where he establishes that: "The more frequently a notion and combination of notions occur, the more importance the author attaches to them as reflecting the essence of his overall idea." (Luhn, 1957, p. 315)

Using this idea, we derive the commonly used assumption below, often referred to as the Luhn Assumption.

Definition 2.5 (Luhn Assumption)**.**

Given a document j, the significance (weight) of a given term i is proportional to the term frequency $f_{i,j}$.

To represent a document by its term frequencies, I introduce the following definition (similar to Baeza-Yates and Ribeiro-Neto (1999)) [1]:

[1]For several of the citations of Baeza-Yates and Ribeiro-Neto (1999) to be found in this section of my paper, I adapted Baeza-Yates and Ribeiro-Neto's definitions and ideas to fit the mathematical-statistical context of this paper.

Definition 2.6 (Term-document matrix)**.**
Let $f_{i,j}$ be the frequency with which term i appears in document j.
The term-document matrix is defined as: $(f_{i,j})_{\substack{i\in I\\ j\in J}}$

where I is the vocabulary size and J is the number of documents[2].
Each column of the matrix corresponds to the j^{th} document vector.

2.2.2 Example

We consider the following three simplified documents to illustrate the term-document matrix:

$$\begin{aligned} d_1 &= \text{"Dog Cat Dog Cat Dog Cat Dog Cat"} \\ d_2 &= \text{"Dog Cat"} \\ d_3 &= \text{"Heteroscedasticity"} \end{aligned} \quad \Rightarrow \quad \begin{pmatrix} 4 & 1 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2.2.3 Term Frequency (TF)

While the mere term frequency $f_{i,j}$ can be used as a term frequency weight, other transformations can be considered, such as a boolean (i.e. 1 if term i occurs in document j, 0 otherwise) or more complex forms, such as a *log*–transformation of the term frequency, which is also proposed by Baeza-Yates and Ribeiro-Neto (1999) and very commonly found in other publications as well:

Definition 2.7 (Term frequency)**.**

$$tf_{i,j} = \begin{cases} 1 + \log f_{i,j} & \text{, if } f_{i,j} > 0 \\ 0 & \text{, otherwise} \end{cases}$$

2.2.4 Inverse Document Frequency (IDF)

In a given document of a given corpus, some terms will have more importance than others when comparing documents. While some of these will usually already be filtered by stop word lists (see Section 2.1.2), such as "the", "a", etc, there is also a gradual distinction in word importance in respect to understanding the nature of the given document.
A term like "martingale" carries a lot more information about the nature of the document (e.g. a subject likely in close proximity of the field *stochastic processes*) opposed to the term "therefore", which does not let us detect similar subjects. To reflect this human intuition in

[2]We will later use N instead of J since it is more common in literature.

information retrieval models, we need a second component to our *term frequency* that takes the rarity of the term into account.

Definition 2.8 (Document Frequency)**.**
Given a corpus of size N[3].
The *Document Frequency*, denoted df_i, of a given term i defines the number of documents in which the given term i appears in.

Evidently, the following property holds: $\forall i \in \mathbb{N}, i \leq I : \; df_i \leq N$

Definition 2.9 (Zipf's Law and distribution)**.**
Let z be a function, $z : \mathbb{N} \longrightarrow \mathbb{N}$, such that $z(r)$ maps to the frequency of the rth most frequent term.
The frequency $z(r)$ is distributed as follows (Baeza-Yates and Ribeiro-Neto, 1999):

$$z(r) = r^{-\alpha}$$

where α is an empirical constant, that approximates 1 for the English language (Baeza-Yates and Ribeiro-Neto, 1999).

Zipf's Law belongs to the family of *power laws*, which are distributions where one quantity changes as the power of another quantity, hence the change of one variable is a relative proportion of that change to another.
While Zipf's Law already serves a purpose by itself for understanding word distributions, we can use it to derive the following definition (as shown in Murphy (2012, Chapter 2)):

Definition 2.10 (Inverse Document Frequency)**.**
Given a corpus of size N.
The *Inverse Document Frequency* of the term $i \leq I$, denoted idf_i, is defined as

$$idf_i = \log \frac{N}{n_i}$$

After having explored the two components of the *tf idf* weighting, we arrive at the following rather trivial definition, which nevertheless is a very prominent concept in information retrieval:

Definition 2.11 (*tf idf* weighting)**.**
Given a corpus of size N.

$$\forall i \leq I : \quad tfidf_i = tf_i \times idf_i$$

[3] N refers to the number of document contained in the corpus

2.2.5 *tf idf*: A Probabilistic Perspective

Clearly, *tf idf*, as defined above, is a statistic. Over the past decades it has been subject to numerous applications and studies, however, *tf idf* was a priori not rooted in well-developed probabilistic reasoning, as Hiemstra (2000) points out. Results using this weight were superior to alternatives, which was many times sufficient justification for its use in information retrieval.

Following Hiemstra's outline, to understand the probabilistic background of TF-IDF, we reconsider the original problem that TF-IDF solves: Determine a numerical measure for how likely a given textual event T is, given the corpus $\mathcal{D}$, i.e. $P(T|\mathcal{D})$. The biggest challenge in natural language processing is that this numerical statistic has to face very sparse data, due to the fact that a document will generally use a small proportion of the vocabulary available. In conclusion, a simple maximum likelihood estimation is not suited and an approach that is not affected by zero entries is needed (Hiemstra, 2000).

Hiemstra postulates a linear interpolation - combining two probability estimates - in the shape of

$$P(T|\mathcal{D}) = \alpha_1 P_{MLE}(T) + \alpha_2 P_{MLE}(T|\mathcal{D})$$

which, using document properties, can be used as follows:

$$\begin{aligned} P(T_i = t_i|\mathcal{D} = d) &= \alpha_1 \frac{df(t_i)}{\sum_t df(t)} + \alpha_2 \frac{tf(t_i, d)}{\sum_t tf(t, d)} \\ &= 1 + \frac{1}{df(t_i)} \times tf(t_i, d) \times \frac{\alpha_2 \sum_t df(t_i)}{\alpha_1 \sum_t tf(t, d)} \end{aligned} \tag{2.1}$$

Taking into account the constant factors, we can directly interpret this as a TF IDF weight. (Hiemstra, 2000)

2.2.6 Visualization of TF-IDF using Python on Wikipedia

Using Python with several extensions, I use the English Wikipedia (see Section 5.1) to analyse the document frequencies (df) of 100 000 occurring index terms. Sorting the index terms by document frequency, I created a plot that represents the first 1000 index terms and their document frequency. For illustration purposes only every 100th word is displayed on the axis. For example the word “according” appears in 407 645 articles of the English Wikipedia; the word “source” appears only in 179 758.

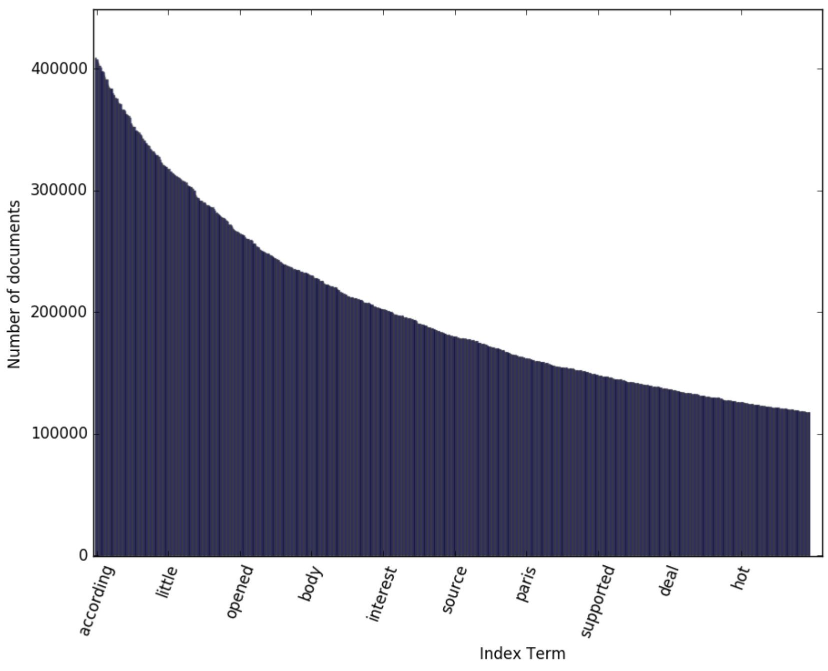


Figure 1: Document Frequency (DF) for 1 000 of the 100 000 ordered index terms generated through 4.1 million Wikipedia articles. [*VisualTFIDF*]

2.2.7 Document Similarity

In text mining, it is generally of interest to evaluate the similarity of documents or topics. While a range of different distance measures is available, not all are equally suited for document comparison.

2.2.8 Euclidean norm

Given an n-dimensional (not necessarily normalized) document space, a first choice might be the well-established Euclidean distance, which simply measures the shortest line between two points (e.g. document representations). A Euclidean norm is in fact the L_2-norm. Further details on the Euclidean norm can be found in Härdle and Simar (2012, p. 335 ff).
We define the Euclidean distance $s(\cdot)$ as follows:

Definition 2.12 (Euclidean distance)**.**
Let $d_1 = (d_{1,1}, d_{1,2}, ..., d_{1,D})$ and $d_2 = (d_{2,1}, d_{2,2}, ..., d_{2,D})$ be the vector representation of two documents of equal length D.

$$s(d_1, d_2) = \sqrt{(d_{1,1} - d_{2,1})^2 + ... + (d_{1,D} - d_{2,D})^2}$$

2.2.9 Limitations of Euclidean distance in document spaces

Using an intuitive example I point out why the Euclidean distance is not well suited for document comparison in the vector space model:
Let d_1, d_2 and d_3 be strongly simplified documents, defined as follows:

$$\begin{aligned} d_1 &= \text{"Dog Cat Dog Cat Dog Cat Dog Cat"} \\ d_2 &= \text{"Dog Cat"} \\ d_3 &= \text{"Heteroscedasticity"} \end{aligned} \tag{2.2}$$

Using a three-dimensional document space, we can represent the documents as follows:

$$\vec{d_1} = (4,4,0),\ \vec{d_2} = (1,1,0),\ \vec{d_3} = (0,0,1)$$

Using definition (2.12), we receive the following distances

$$\begin{aligned} s(\vec{d_1}, \vec{d_2}) &\approx 4.2 \\ s(\vec{d_1}, \vec{d_3}) &\approx 5.74 \\ s(\vec{d_2}, \vec{d_3}) &\approx 1.73 \end{aligned}$$

Hence, the Euclidean distance suggests that documents 2 and 3 are a lot more similar than documents 1 and 2, which strongly conflicts our human understanding of document similarity.

2.2.10 Cosine Similarity

In their research published in 1975, Salton, Wong, and Yang postulate the cosine similarity as a useful measurement of document similarity, which has since then become a very commonly used distance measure for document spaces.

Definition 2.13 (Cosine similarity)**.**
Let $d_1 = (d_{1,1}, d_{1,2}, ..., d_{1,D})$ and $d_2 = (d_{2,1}, d_{2,2}, ..., d_{2,D})$ be the vector representation of two documents of equal length D. We define the cosine similarity between document 1 and document 2 as:

$$s(d_1, d_2) = \frac{d_1 \cdot d_2}{\|d_1\| \, \|d_2\|}$$

where $\| \circ \|$ is the Euclidean norm on the D-dimensional document space, defined as $\|x\| = \sqrt{x_1^2 + ... + x_D^2}$.

This definition brings the following consequences:
Since for all documents d_i with $i \in \{1, ...N\}$ we restrict $\forall j \in \{1, .., D\} : d_{i,j} \geq 0$ – i.e. documents cannot experience negative word counts – the angle between two document vectors is restricted to at most 90 degrees, hence $\forall(i, j) \in \mathbb{N}^N \times \mathbb{N}^D : s(d_1, d_2) \in [0, 1]$, i.e. document distances cannot be negative and can be at most 1.

2.2.10.1 Evaluation For cosine similarity, the length of the vector does not impact the distance measure, instead the relative distribution of words matters. Reusing the example (2.2), we receive the results:

$$s(\vec{d_1}, \vec{d_2}) = 1$$
$$s(\vec{d_1}, \vec{d_3}) = 0$$
$$s(\vec{d_2}, \vec{d_3}) = 0$$

Evidently this represents well the context.

2.3 Visualization: Classic literature in a vector space

The famous works of Leo Tolstoy, Jane Austen and James Joyce allow for more than just a great enriching literary journey: Having introduced the vector space model, we can represent the documents in a vector space, open for analysis under different perspectives.
For this example I use the three works: (1) *War and Peace* by Leo Tolstoy (1869), (2) *Pride and Prejudice* by Jane Austen (1813), and (3) *Ulysses* by James Joyce (1922), which I accessed using the *Gutenberg Project*'s website[4].

After having removed everything but letters and numbers - about 900 000 words remain - I used the Python programming language to process the documents into a data frame containing only the unweighted term frequencies. Since our eyes are restricted to 3 dimensional spaces, I chose three representative index terms ('God', 'time' and 'love'), however, there are numerous possibilities - the three works together experience several thousand distinct index terms. The term occurrences were then normalized using each document's word count. Advanced techniques of projection were not applied in this case.

[4]https://www.gutenberg.org/

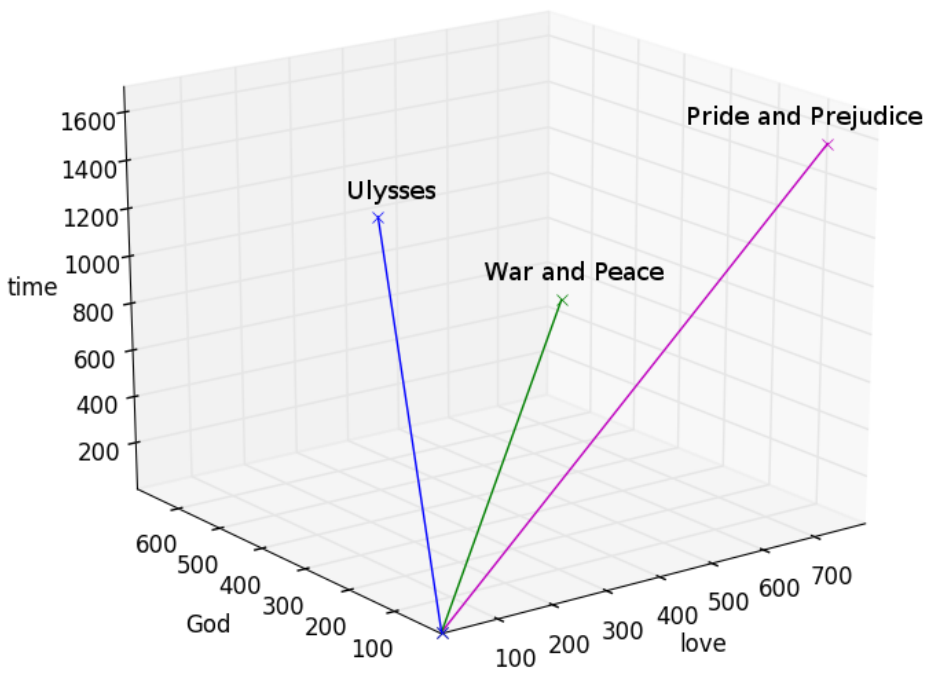


Figure 2: Representation of literature works in a vector space of the three words 'God', 'time' and 'love' [*Doc2VectorSpace*]

3 Statistical Concepts

3.1 Latent Variables

The concept of latent variables has been applied in a vast amount of fields, ranging from social sciences and psychology to economics. In all fields, we find that latent variables allow us to express concepts that are not observable but of high interest to the model. Everitt (1984) presents the examples of social class and intelligence. While both are not observable, their impact is.

The *Latent Dirichlet Allocation* model, discussed in Section 4, already carries the term *Latent* in its name, however, latent variables play a significant role in most models of natural language processing, not only LDA. In natural language processing we encounter latent variables often in the form of a *topic*.

I illustrate this with the following example: Given a corpus of the two simplified documents "Webcam, Keyboard, Monitor" and "Cat, Dog, Hamster". We recognize without hesitation that these two documents are about two different topics, for example "computer accessories" and "domestic animals / pets". These topics are the latent variables behind the text generation in our imagined model. Each topic is associated to different words. However, in some cases it might be necessary to allow for topics to share different index terms. In our case, this can be visualized by adding the token *Mouse* to both documents. *Mouse* could be generative for the topic “pet” or equally for the topic “computer accessory”. Some models in natural language take this into account.

Aside of topic models, a widely used statistical tool that extracts latent variables is for example *Factor Analysis*, where the latent variables are called factors. This technique cannot be elaborated in this paper but is explained in detail in Härdle and Simar (2012, Chapter 10).

3.1.1 Latent Class Model

A *Latent Class Model* is a type of *Latent Variable Model*, which is a family of models that picture correlations between variables as arising from latent variables (Murphy, 2012).
In most models we follow the assumption that the number of visible variables is a lot larger than the number of latent variables. However, certain models will differ.

3.2 Exchangeability as of de Finetti

In 1931 Bruno de Finetti developed a theorem, usually referred to as *de Finetti's Theorem of Exchangeability* in which he states:

Definition 3.1 (Exchangeability)**.**
A binary sequence $X_1, X_2, ...$ is exchangeable if and only if $X_1, X_2, ..$ are mixtures of Bernoulli sequences (Lauritzen, 2007).

Since in most cases topic models handle finite amounts of documents, we will be in need of a finite derivation of de Finetti's theorem, similar to the one used by Blei et al.:

Definition 3.2 (Finite Exchangeability)**.**
Let $X_1, ..., X_n$ be a sequence of random variables, e.g. documents. Let P denote the random variables' joint probability distribution.
The given sequence $X_1, ..., X_n$ is *n-exchangeable*, if for any finite permutation $X_{\pi(1)}, ..., X_{\pi(n)}$ the joint probability distribution remains unchanged, i.e.

$$P(X_1, ..., X_n) = P(X_{\pi(1)}, ..., X_{\pi(n)})$$

It is important to make a distinction here: Exchangeability does not necessarily imply *independent and identically distributed* (iid). While they are indeed identically distributed, they are only conditionally independent (Blei et al., 2003).

3.3 Dirichlet Distribution

3.3.1 Definition

In the following, I use the notation and choice of variable names in accordance with Blei et al. (2003).
To define the Dirichlet distribution, I introduce at first the Gamma and Beta function, also known as Euler integral of the second kind and Euler integral of the first kind respectively.

Definition 3.3 (Gamma and Beta Function)**.**
The following two definitions can be found similarly in Jeffrey (2004).
Gamma function:

$$\forall \alpha \in \mathbb{R}_+ : \qquad \Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} \, dx \tag{3.1.1}$$

Beta function:

$$\forall (\alpha, \beta) \in \mathbb{R}_+^2 : \qquad B(\alpha, \beta) = \int_0^1 x^{\alpha-1} (1-x)^{\beta-1} \, dx \tag{3.1.2}$$

Corollary 3.1.
It can be shown that the Beta Function can also be represented as a function of Gamma Functions as follows:

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

(Murphy, 2012)
We use this expression to extent the beta function from 2 arguments to a multivariate beta function with k arguments.

$$B(\alpha_1, ..., \alpha_k) = \frac{\Gamma(\alpha_1) \times ... \times \Gamma(\alpha_k)}{\Gamma(\alpha_1 + ... + \alpha_k)} = \frac{\prod_{i=1}^k \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^k \alpha_i\right)} \tag{3.1}$$

I introduce the k-dimensional Dirichlet distribution by its probability density function:

Definition 3.4 (Dirichlet Distribution)**.**

$$p(\theta|\alpha) = \frac{1}{B(\alpha_1, ..., \alpha_k)} \, \theta_1^{\alpha_1 - 1} ... \, \theta_k^{\alpha_k - 1} \tag{3.2}$$

where $B(\alpha_1, ..., \alpha_k)$ is the multivariate Beta function, defined above, and θ, α, two k-dimensional vectors.

A probability density function, $p(\theta)$ must per definition satisfy the following condition:

$$\int_{-\infty}^{+\infty} p(\theta) = 1$$

In the case of the Dirichlet distribution, to guarantee this condition, the factor $\frac{1}{B(\alpha_1,...,\alpha_k)}$ functions as normalizing constant. A k-dimensional Dirichlet distribution lives on a $k-1$-dimensional simplex, since its support is $\sum_i x = 1$ (Murphy, 2012). This is a key aspect to the usage in topic models, as we will see later.

3.4 Bayesian concepts

Bayesian statistics and frequentism, though being able to apply majorly the same techniques, differ in a few significant points that eventually make them more or less suited for their application in machine learning, such as natural language processing. In the past, Bayesian statistics have emerged to be commonly found in machine learning and equally in the probabilistic models described later.
A few key concepts and aspects useful to the development and understanding of probabilistic topic models shall be introduced hereafter.
A major difference is the point of view on the essence itself: The concept of probability. Frequentists see a probability as an objective characteristic – a fact – given by the object of interest itself, which also accounts for parameters, which are fixed constants. Bayesian statistics, in contrast, propagate a subjective, belief-based probability and parameters expressible as probability statements (Wasserman, 2010). For example, this is imposed by the concept of priors as we will see below. Bayesian statistics, hence, provide a looser framework.

3.4.1 Prior distribution

As mentioned above, a prior probability distribution generally expresses a belief about a parameter before the data has been taken into account to gain information on the true parameter. Priors can have different natures, depending on the context.
A non-informative prior is one that is at the state of knowledge the most objective possible and usually bears only general knowledge. This will often be in the form of a uniform distribution. For example $p(\theta) \propto \textit{constant}$. Non-informative priors can also be more sophisticated, such

as *Jeffreys Prior*, which is based on the Fisher information matrix, related to the Kullback–Leibler divergence (see Section 3.6).
A given prior is called *conjugate prior for the corresponding likelihood function* if the prior and the posterior belong to the same family of probability distribution (Murphy, 2012). Having a conjugate prior simplifies the computation of the posterior and therefore finds wide application in machine learning. A simple view on Bayesian statistics is that

$$\text{Posterior Probability } \propto \text{Prior Probability} \times \text{Likelihood}$$

More precisely:

Definition 3.5 (Posterior Probability distribution)**.**
Let θ be the parameter of interest and $\mathcal{D}$ the given data. We define the posterior distribution conditional on the data:

$$p(\theta|\mathcal{D}) = \frac{p(\theta) \times p(\mathcal{D}|\theta)}{\int p(\mathcal{D}|\tilde{\theta})p(\tilde{\theta})\, d\tilde{\theta}}$$

where $\int p(\mathcal{D}|\tilde{\theta})p(\tilde{\theta})\, d\tilde{\theta}$ is the normalizing constant.
(similar in Wasserman (2010))

If prior and likelihood are conjugates, then we can write the posterior as a closed formula. As the data increases, the prior plays a less and less significant role, which is one of the reasons why it finds a lot of use in machine learning.

3.5 Model fitting

3.5.1 Overfitting

A common problem in machine learning is *overfitting.* That is, when fitting the statistical model using a training set, we want to be highly aware of the issue of modeling too precise with respect to the given training set. When modeling every aspect for example, the model will instead memorize the data, making predictions of so far unseen data impossible.
An intuitive approach to keep overfitting under watch is a count based on the successes of predictions, called *misclassification rate* (Murphy, 2012) or *training error rate* (Wasserman, 2010), defined as follows

Definition 3.6 (Misclassification rate)**.**
Given a labeled training set of size N: $\mathcal{T} = \{x_i, y_i\}_{i=1}^{N}$ with (x_i, y_i) being a tuple of input

and output respectively. Given a classifier function $f(x_i)$.
We define the misclassification rate as:

$$mr(f, \mathcal{T}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(f(x_i) \neq y_i)$$

This approach is intuitive but clearly limited.
To rule out that our model just memorized (overfitted) the training set, we split the data available (and originally used as training set) into a part that remains in use as a training set, and a part that becomes the *validation set.* Murphy (2012) presents a commonly used proportion of 80% training set and 20% validation set.
Murphy remarks that also this approach is prone to error depending on the structure of the given data, namely if splitting the available data makes the training set too small for a proper fitting procedure. An alternative approach that might be the solution is called *Cross Validation.* Due to the focus of this work, I cannot outline this section further and therefore suggest Chapter 6.5 of Murphy (2012) for further reading concerning cross validation and other approaches.
Putting a metric of error for a given training set will only work for supervised learning, because unsupervised learning lacks the possibility to verify the results in such a way.

3.5.2 Visualisation using Decision trees

Using Python I want to demonstrate visually how overfitting can look like, using a *Decision Tree Regression.* I construct a data set as follows:

I generate a random set of x values in a given range, and draw y values with strong Gaussian noise:

$$y_i = x_i^3 + \varepsilon_i \quad \text{with} \quad \varepsilon_i \sim \mathcal{N}(0, 400)$$

I run a decision tree regression through the data points, once with depth 3 and once with depth 10. We see that depth 10 tends to overfit. For the decision tree I use the commonly used machine learning library *scikit-learn.*

3.6 Kullback-Leibler divergence

Many probabilistic topic models aim at minimizing dissimilarity between the empirical probability distribution and the model's probability distribution. From this arises the need for a measure of dissimilarity.

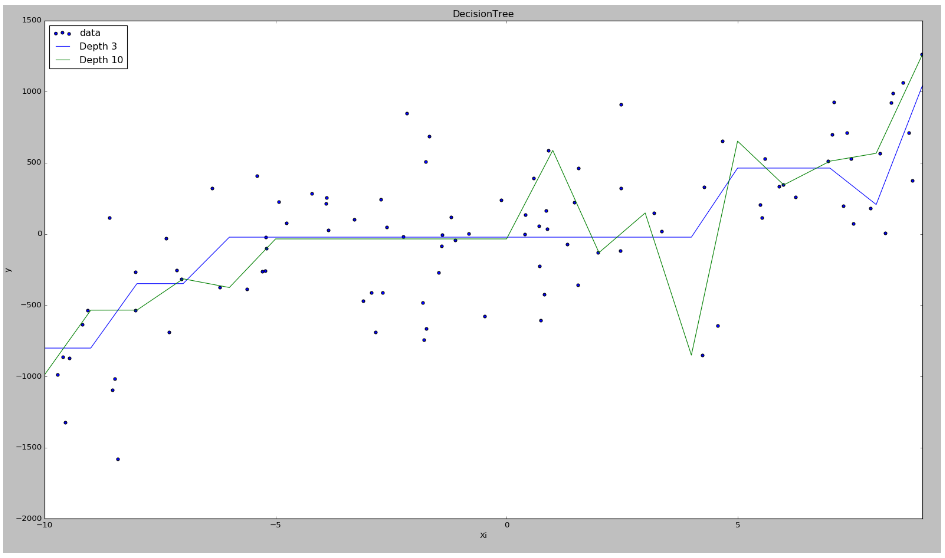


Figure 3: Visualization of overfitting using a decision tree algorithm on a generated data cloud. [*VisualOverfittingExemp*]

One very common tool used in information theory, is the referred to as *Kullback-Leibler divergence* (KL-divergence), also sometimes called relative entropy.

Definition 3.7 (Kullback-Leibler divergence / relative entropy)**.**
Let P and Q be two random variables. Let p and q be the corresponding probability distributions.
The Kullback-Leibler divergence for P and Q discrete is defined as:

$$\mathbb{KL}(P||Q) = \sum_{i=1}^{n} p_i \, \frac{p_i}{q_i} \, dx$$

The Kullback-Leibler divergence for P and Q continuous is defined as

$$\mathbb{KL}(P||Q) = \mathbb{E}\left[\log \frac{p(x)}{q(x)}\right] = \int_{-\infty}^{\infty} p(x) \, \log \left(\frac{p(x)}{q(x)}\right) \, dx$$

(similar in Murphy (2012) and Wasserman (2010)

A famous "family member" of the KL-divergence is the *Fisher Information*, also commonly used in statistics. The Fisher Information, noted $I(\theta)$, is in fact the variance of the score, noted $s_\theta(X)$, which in turn is a numerical measure for the sensitivity of a log-likelihood: $s_\theta(X) = \frac{\partial}{\partial\theta} \log L(X, \theta)$

Definition 3.8 (Fisher Information)**.**

$$I(\theta) = \mathbb{V}ar_\theta \left[s_\theta X\right] = -\mathbb{E}_\theta \left[\frac{\partial^2}{\partial\theta\partial\theta'} \log L(X; \theta)\right]$$

(Wasserman, 2010)

The relation between Kullback-Leibler divergence and Fisher information is as follows: The Hessian of the KL-divergence is the Fisher information metric

$$I(\tilde{\theta}) = \frac{\partial^2}{\partial\theta\partial\theta'}\bigg|_{\theta=\tilde{\theta}} \mathbb{KL}(\tilde{\theta}||\theta)$$

This property can be shown directly by calculating the Hessian of $\mathbb{KL}(\tilde{\theta}||\theta)$.
It has to be noted that the Kullback-Leibler divergence has limitations that need to be obeyed in application: The KL-divergence is not a distance function in the mathematical sense as it lacks the two critical requirements for a metric: (1) symmetry and (2) triangle inequality. An adaptation of the KL-divergence that fulfills the criteria of a metric is called the *Jensen-Shannon divergence.*
Furthermore, the KL-divergence is inapplicable for probability vectors containing zeros, which can be derived directly from the definition. This can be an issue in topic modeling, since sparse vectors (with many zero entries) occur often.

3.7 Methods of Inference

The variety of machine learning algorithms also faces a large variety of inference methods that can be applied. This section presents a selection of inference methods that have a strong relation with the topic models presented in Section 4. The difficulty of inference concerning the posterior distribution of probabilistic topic models is strongly dependent on the type of prior chosen. If the prior is conjugate to the likelihood we can compute the posterior in its closed form. In other cases this might not be possible. Since this difference is a major distinction between the two later presented models *Latend Dirichlet Allocation* and *Dynamic Topic Model*, I present hereafter methods for both tasks.

3.7.1 Variational Bayesian Inference and Tractable[5] Substructures

Variational inference is used when facing a non-tractable probability distribution that we want to approximate. The algorithm begins by choosing a tractable probability distribution $q(x)$ of simpler form than the distribution to be approximated. The inference problem becomes an optimization problem: Minimize the dissimilarity between $q(x)$ and the true posterior distribution $p^*(x)$.

KL-divergence (see above: 3.6) is a possible and very common choice, though still needs to be used with caution: While $\mathbb{KL}(p^*||q)$ is intuitive, this would lead to computational problems,

[5]A problem is said to be *tractable* if it can be computationally solved in polynomial time

while the reverse $\mathbb{KL}(q||p^*)$, also referred to as *reverse KL*, allows for a computation, which is elaborated in (Murphy, 2012).

Mean-field variational Bayes is a type of variational inference where KL-divergence is used for dissimilarity measures. Furthermore, we assume that the posterior is made up of independent variables, i.e. *Complete Factorizability.*
Saul and Jordan (1995) present an adaptation of this method, which relaxes the restriction of complete factorizability and allows for a grouping of variables in *tractable substructures.* This method is referred to as *structured mean field* and can be applied to the later presented *Latent Dirichlet Allocation* model.

3.7.2 (Variational) Kalman Filtering

The *Kalman Filter* is an algorithm that is very commonly applied in location systems and system controls, for example for determining the velocity and position of some sort of vehicle. While this seems far away from topic models, Kalman filters are very powerful tools and are well suited for many environments that concern measurements of variables with statistical noise and hence find application in many machine learning algorithms.
This algorithm processes data recursively and takes into account all information available. This information may contain measurement errors - generally assumed to be Gaussian - for which the Kalman filter will minimise the mean square error of the parameters to be estimated. Picking up the terminology of Bayesian statistics, the objective that the Kalman filter pursues is to determine the conditional probability density of the variables to be estimated. This is conditioned on the information derived from the actual data, for example through measurement (Maybeck, 1979).
The algorithm operates in two steps: (1) *Prediction* and (2) *Measurement* - the latter sometimes also referred to as *Update.*
Let θ_t be a latent variable and y_t the observation at time t. We note c_t a control signal, not mandatory in each state. The observation is a function of the latent variable.
(1) Prediction:
Predict the marginal posterior of the variable θ at time t, using the observation and control signal

$$p\left(\theta_t|y_1, ..., y_{t-1}, u_1, ..., u_t\right) = \mathcal{N}\left(\theta_t|\mu_{t|t-1},\ \Sigma_{t|t-1}\right)$$

(2) Measurement:
Using weights that enforce the observations with higher certainty, the next observation is

measured. We recall Bayes rule from above (Section 3.4) and interpret the prediction like a prior, resulting in

$$p(\theta|y_1, ..., y_{t-1}, y_t; u_1, ..., u_t) = \underbrace{p(\theta_t|y_1, ..., y_{t-1}, u_1, ..., y_t)}_{\text{prediction}} \times \underbrace{p(y_t|\theta_t, u_t)}_{\text{likelihood}} = \mathcal{N}(\theta_t|\mu_t,\ \Sigma_t)$$

where Σ is the covariance matrix. For a detailed derivation, especially concerning the parameters of the normal distribution, see Murphy (2012, Chapter 18).

Observation counts on a very large scale, such as those commonly found in machine learning applications, let standard Kalman filters face intricate challenges. The parameters of the normal distribution are matrices that grow with the number of steps. Since the Kalman filter computation demands for matrix multiplication and inversion, computation time will reach critical values with very large observation counts.
The *Variational Kalman Filter* (VKF) method addresses exactly this issue. Here, filter estimates and its covariance are computed using an iterative minimization method. This makes it less memory demanding and hence suited for inference of machine learning algorithms. such as the *Dynamic Topic Model*.

3.7.3 Gibbs Sampling

A *Gibbs sampler* is a method based on *Markov Chain Monte Carlo* (MCMC), which is a class of algorithms where sampling from a probability distribution is conducted using a Markov chain, for which the desired probability distribution will be reached in the equilibrium. Gibbs sampling is a special case of the *Metropolis-Hastings algorithm*, a MCMC algorithm that was developed in the 1950's.
Chen et al. (2000) give a detailed introduction to Gibbs sampling, which I present slightly modified:
We work on a K-dimensional parameter vector $\theta^{(i)} = (\theta_1^{(i)}, ..., \theta_K^{(i)})$ and want to determine its true parameters by sampling. We note θ_{-j} the vector θ without the jth value. We note $\mathcal{D}$ the given data.
We start with a random starting point, where $i = 0$:

$$\theta^0 = \left(\theta_1^0, ..., \theta_K^0\right)$$

Now we generate each new $\theta^{(i+1)}$, written in algorithmic form:
For each i:

For each j:

$$\theta_j^{(i+1)} \sim p(\theta_j | \theta_{-j}, \mathcal{D})$$

Here $p(\theta_j|\theta_{-j})$ is called the *full conditional* for variable j (Murphy, 2012).
Since we start at a random start point, the first samples will not yet be of the probability distribution that we desire to approximate. Due to this, it is common to discard a portion of the samples, called the *burn-in phase.* This could be for example the first 25%.

3.7.4 Grouped and collapsed Gibbs Sampling

After having a notion of Gibbs sampling we adapt the sampling method such that we can benefit from knowledge of the given variables to make the algorithm more efficient.
Grouped Gibbs sampling, also called *blocked Gibbs sampling*, takes two or more elements from the vector θ to sample them together. Thus for $m, n \in \{1, ..., K\}$, we replace:

$$\theta_m \sim p(\theta_m | \theta_{-m}, \mathcal{D})$$

$$\theta_n \sim p(\theta_n | \theta_{-n}, \mathcal{D})$$

with

$$(\theta_m, \theta_n) \sim p(\theta_m, \theta_n | \theta_{-(m,n)}, \mathcal{D})$$

(θ_m, θ_n) are simultaneously drawn from their joint probability distribution.
Grouped Gibbs sampling decreases the number of steps the algorithm has to take in the state space, when correlation is high (Murphy, 2012).

We can further adapt the Gibbs sampler to create the *Collapsed Gibbs sampler.* Collapsed refers in this case to the fact that we integrate out a given number of variables, naturally reducing the dimensions of the sample space.
For m and n, we replace:

$$\theta_m \sim p(\theta_m | \theta_{-m}, \mathcal{D})$$

$$\theta_n \sim p(\theta_n | \theta_{-n}, \mathcal{D})$$

with

$$(\theta_m, \theta_n) \sim p(\theta_m, \theta_n | \mathcal{D})$$

As we see, the difference between grouped and collapsed Gibbs sampling is that for θ_m and θ_n we do not draw from full conditional with respect to $\theta_{-(m,n)}$ anymore but only from the

marginal distribution.
(Chen et al., 2000).

As Murphy (2012) points out, samples drawn through collapsed Gibbs sampling benefit from a much lower variance due to them not being drawn from the entire joint state space, increasing the method's efficiency.

3.7.5 Method choice

While some of the above methods are interchangeable, there are differences in their assumptions and performance, also limiting their use in certain cases.
One aspect to consider when choosing the appropriate inference method is the size of the data set. Variational inference is fast for small and medium observation counts, while Markov Chain Monte Carlo is especially well suited for very large data sets.
Also the type of prior used can prohibit or at least complicate the use of certain inference methods. As Blei and Lafferty point out, Gibbs sampling performs well on static topic models but is strongly challenged when it comes to dynamic models with non-conjugate priors.
Of course an easy implementation can also favor one inference method over the other. Many Markov Chain Monte Carlo methods benefit from easy implementation.

4 Probabilistic Topic Models

While this chapter aims at presenting the Latent Dirichlet Allocation (LDA) model and the Dynamic Topic Model (DTM), understanding their key aspects also makes it necessary to understand the development of the preceding models, namely *Latent Semantic Analysis* (LSA) and *probabilistic Latent Semantic Analysis* (pLSA), which I will therefore also present in a compact version in the following sections.
As mentioned above, Probabilistic Topic Models are not restricted to its use in language processing, however, it will be restricted to this main field in this section.

Probabilistic Topic Models generally work on a high-dimensional document space in which topic modeling algorithms discover the latent topic structure of the given document corpus (compare Blei (2012)). The models work on the basis of the observed variables, namely the given words per document.

4.1 Model types

4.1.1 Generative models

A generative model uses its parameters, generally including latent parameters, to construct the observable variables. In the case of text data, these observable variables are the words in the documents, while the latent variables are the topics. These topics are in most cases not labelled (i.e. entitled by a human), which is characteristic for unsupervised learning. *Latent Dirichlet Allocation* and *Dynamic Topic Models*, explained below, are both examples for generative models.
A generative model establishes the joint probability distribution and hence, it is often an intermediate step to understanding the nature of how the given documents were formed. Using inference methods, the true parameters of the model are estimated.

4.1.2 Mixed membership models

For many clustering models we find that data points can only be associated to one latent variable. While this is convenient in many examples, such as using machine learning to distinguish pictures of dogs and cats, evidently this conflicts with our human understanding of documents, since we find that documents can indeed contain a range of topics.
A mixed membership model contrasts these standard clustering techniques in the way that it allows data points to belong to numerous categories in varying degrees. In the sense of

documents, this allows a document to be associated to numerous topics in various proportions, as our human understanding of documents agrees upon. For this reason topic models and mixed membership models have a strong link.
Airoldi et al. (2014) show many different applications of mixed membership models in detail.

4.2 Latent Semantic Analysis (LSA)

Latent Semantic Analysis is a technique patented by Deerwester et al. in the year 1988. In LSA, a dimensionality reduction projects the documents onto a so called latent semantic space with the goal of finding semantic relations between the given entities (e.g. the documents in the corpus)(Hofmann, 1999).
It is based on the concept of the vector space model, explained in Section 2.2: A large term-document matrix is constructed and tf-idf weights assigned (other methods are possible). Mathematically, the method of LSA is based on *Singular value decomposition*, a factorization technique from linear algebra, where a $L2$ norm determines the optimal decomposition (Hofmann, 2001).

4.3 Probabilistic Latent Semantic Analysis (pLSA)

In 1999, Hofmann introduced an extension for the previously developed Latent Semantic Analysis (LSA) by distancing the model from algebraic approaches and building it on a statistical foundation, namely a latent class model (see Section 3.1.1).
To determine the optimal decomposition in pLSA, the singular value decomposition is omitted and instead minimises the Kullback-Leibler divergence between the empirically derived probability distribution and the model's probability distribution (Hofmann, 2001), as explained in Section 3.6.

As we will see in the following, the *Latent Dirichlet Allocation* model is in principle a probabilistic extension of *LSA* with *pLSA* being a close predecessor to *LDA* (Murphy, 2012, Chapter 27), where Blei addresses some of the shortcomings of pLSA.
One of pLSA's weaknesses is its proneness to overfitting (Blei and Lafferty, 2006). Hofmann already attempts to address the issue by using a so called *Tempered Expectation-maximization* (TEM) algorithm, but the fact that pLSA's parameters grow linearly with the number of documents makes overfitting more likely (Hofmann (1999), Blei et al. (2003)).

4.4 Latent Dirichlet Allocation (LDA)

The Latent Dirichlet Allocation (LDA) is a probabilistic generative model and mixed membership model that was introduced by Blei et al. in the year 2003, which discovers latent topics in large text corpora. It has since become a very commonly used model in natural language processing but also other areas, further elaborated at the end of this paper.
As generative model and mixed membership model implies, LDA postulates that each document arises from a mix of documents in various degrees. An (approximative) illustration is the following example: The model determines that document #1729 is 37% about *topic 17*, 25% about *topic 11*, 14% about *topic 61*, etc. While *topic 17* itself is distributed over the words: 34% *apple*, 21% *banana*, 4% *orange*, 3.8% *lemon*,...
We can derive that *topic 17* might be about "fruits", but since LDA is unsupervised, this label is neither used nor needed.

4.4.1 Generation process

Precisely, Blei et al. (2003) define the generation of a document in a corpus as follows:
A document has length N, that varies according to a Poisson distribution:

$$N \sim \mathcal{P}(\xi)$$

We now have a document that can be filled with N words (word order doesn't matter due to the Bag-of-words assumption, see definition 2.4), noted w_n. Each word, w_n will be derived from a topic z_n.
In the meanwhile, we generate the topics and their associated words as follows:
A topic z_n is multinomially distributed:

$$z_n \sim Multinomial(\theta)$$

The parameter of that multinomial distribution, θ, is itself generated from a Dirichlet distribution, illustrated in section 3.3.

$$\theta \sim Dir(\alpha)$$

It is important to see that while α is a fixed parameter, θ is generated for each document, representing the fact that documents have different topic distributions.
To fill the document we now draw N words w_n from another multinomial distribution, conditioned on the topic and the parameter β.

$$w_n \sim Multinomial(\beta_z)$$

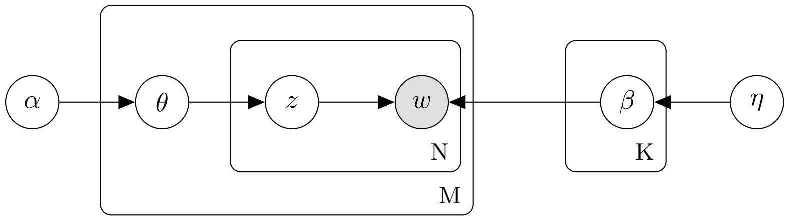


Figure 4: Plate notation of the smoothed LDA with M documents of length N generated through K topics [Created in LaTeX]

Depending on the type of LDA used, namely the original proposal or the smoothed proposal, β is either a fixed parameter or generated through another Dirichlet distribution, respectively. In the case of a smoothed LDA

$$\beta_z \sim Dirichlet(\eta)$$

where η is similarly to α a hyper-parameter for the prior.

4.4.2 Inference

The generative model defines a joint probability distribution. For the (simpler) non-smoothed LDA, where β does not follow a Dirichlet distribution, we have:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta)p(w_n|z_n, \beta) \tag{4.1}$$

(see Blei et al. (2003))

Since we are interested in determining the latent variables, the inference focuses on determining the above written posterior distribution: $p(\theta|\alpha)$.

There are different approaches to infer the parameters. While in their original paper, Blei et al. apply *Variational Bayes* (see section 3.7.1), many authors and programmers apply *Collapsed Gibbs Sampling* (see section 3.7.4), which finds application in numerous other topic models.

For Gibbs sampling, as described in Section 3.7.4, we integrate out certain variables, that do not draw from the full conditional distribution. For a smoothed LDA model, these variables are θ and β - the two variables that are drawn from a Dirichlet prior.

4.4.3 Exchangeability assumption in LDA

In the LDA model, the aspect of exchangeability concerning both, documents and words, the latter being an aspect of the bag-of-words model described in Definition 2.4, has a strong significance for the development of the LDA model.
Blei et al. state that using de Finetti's classical representation theorem we can conclude that any collection of random variables has a representation as a mixture distribution, which founds the reasoning for why LDA needs to be a mixed membership model.
In contrast to the original theorem of de Finetti, our documents and words used in the training set are finite. Therefore we consider the adapted version of de Finetti's theorem, presented in section 3.2.
This idea of exchangeability and the fact that exchangeability implies a mixture distribution is an essential key aspect of the LDA model and causal for its status as a mixed membership model.

4.4.4 LDA as a bag-of-word-model

Using the definition of the bag-of-words model in section 2.4 and simple math, we can show that LDA fulfills the bag-of-words model's key characteristic that word order in a given document does not influence the model.
Keeping in mind the definition of exchangeability, we treat a document as a finite sequence of words, in which case a word is a random variable that draws from a topic with a given multinomial probability distribution p.

We define $w_{d_0,i}$ and $w_{d_0,j}$ as the ith and jth word of a given document d_0 respectively. If LDA is a bag-of-words model, the order of any given words in a document must not matter. Let w_{d_0} be the original vector of words in document d_0 of length N, defined as

$$w_{d_0} = (w_{d_0,1}, ..., w_{d_0,i}, w_{d_0,j}, ..., w_{d_0,N}) \tag{4.2}$$

We now define $\tilde{w}_{d_0} \in \tilde{w}_{1:D}$ as the vector with words $w_{d_0,i}$ and $w_{d_0,j}$ switched (permutation):

$$\tilde{w}_{d_0} = (w_{d_0,1}, ..., w_{d_0,\mathbf{j}}, w_{d_0,\mathbf{i}}, ..., w_{d_0,N}) \tag{4.3}$$

In terms of probability the bag-of-words model is characterized by its joint distribution of all variables to remain unchanged for a change in word order. Hence we want to show that:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, \tilde{w}_{1:D}) \tag{4.4}$$

Using Equation (4.1):

$$\begin{aligned}
p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, \tilde{w}_{1:D}) &= \prod_{i=1}^{K} p(\beta_i) \prod_{\substack{d=1 \\ d \neq d_0}}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(z_{d,n} \mid \theta_d)\, p(w_{d,n} \mid \beta_{1:K}, z_{d,n}) \right) \\
&\quad \times\ p\left(\theta_{d_0}\right) \left(\prod_{n \in \{1,\ldots,\mathbf{j},\mathbf{i},\ldots,N\}} p(z_{d_0,n} | \theta_{d_0})\, p(w_{d_0,n} | \beta_{1:K}, z_{d_0,n} \right) \\
&= \prod_{i=1}^{K} p(\beta_i) \prod_{\substack{d=1 \\ d \neq d_0}}^{D} p(\theta_d) \left(\prod_{n=1}^{N} p(z_{d,n} \mid \theta_d)\, p(w_{d,n} \mid \beta_{1:K}, z_{d,n}) \right) \\
&\quad \times\ p\left(\theta_{d_0}\right) \left(\prod_{n \in \{1,\ldots,\mathbf{i},\mathbf{j},\ldots,N\}} p(z_{d_0,n} | \theta_{d_0})\, p(w_{d_0,n} | \beta_{1:K}, z_{d_0,n} \right) \\
&= p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})
\end{aligned}$$

Evidently Equation (4.1) holds true for any permutation of word order, which describes the bag-of-words model.

4.5 Extensions

Blei's Latent Dirichlet Allocation model has restrictions and assumptions that can be loosened or altered to arrive at a different model with new characteristics.
I present such adaption in the following section.

4.6 Dynamic Topic Models

Blei's Latent Dirichlet Allocation model restricts words to be exchangeable (bag-of-words assumption, see Definition 2.4) but also for document order to be without influence.
In their work of 2006, Blei and Lafferty present an extension of LDA, which relaxes the restriction on the exchangeability of documents, arguing that for numerous sets of documents, namely emails, articles, and such, LDA does not reflect the fact that content generation evolves over time. Thus, Blei and Lafferty develop a model that represents the dynamics of topic changes over time, or in Blei and Lafferty's words a "statistical model of topic evolution".

4.6.1 Sequential data assumption and generation process

In contrast to LDA, DTM groups data by time slices of a fixed duration, such as months or years. We assume that the topics that generate the data (e.g. the text documents) at time t have evolved from the topics at time $t-1$. This small assumption implies strong changes to

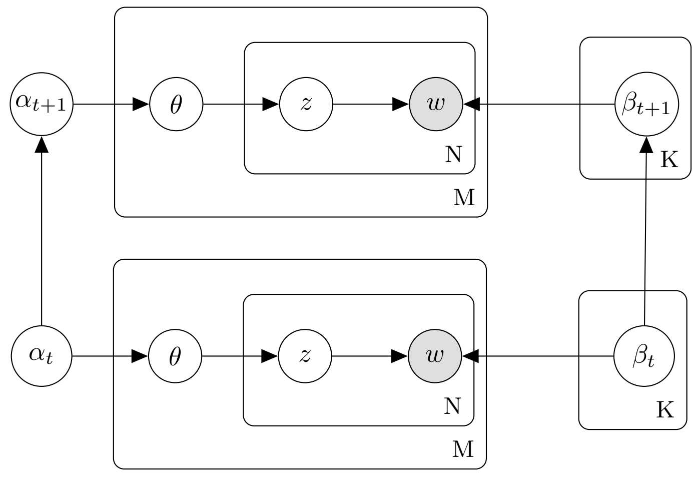


Figure 5: Plate notation of the DTM with M documents of length N, generated through K topics [Created in LaTeX]

the nature of the model. The documents are now generated sequentially, which disqualifies the centerpiece of LDA: The dirichlet distribution. Instead, Blei and Lafferty propose a chain of normal distributions, as follows:

Let $\beta_{t,k}$ be the word distribution for a given topic k in time slice t, i.e. the parameters for how words are generated by a given topic. Let α_t be the topic distribution for a document at time t.

Knowing the word distribution for topic k in the time slice before, namely $\beta_{t-1,k}$, we assume that at time t, the word distribution will remain in expectation the same $\mathbb{E}[\beta_{t,k}|\beta_{t-1,k}] = \beta_{t-1,k}$ with a normal distribution under variance $\sigma^2 I$, referred to by Blei et al. as Gaussian noise.

$$\beta_{t,k}|\beta_{t-1,k} \sim \mathcal{N}(\beta_{t-1,k}, \sigma^2 I)$$

In the same manner, we assume for α_t:

$$\alpha_t|\alpha_{t-1} \sim \mathcal{N}(\alpha_{t-1}, \delta^2 I)$$

In LDA, α and η (here β) are the parameters for the Dirichlet prior of the topic distribution of a document and the word distribution of a topic, respectively. From there on, θ and β are the actual topic distribution of a document and the word distribution of a topic. Since we omit the Dirichlet prior, topic distributions of document d at time t ($\theta_{t,d}$) are now drawn from a normal distribution with mean α: $\theta_{t,d} \sim \mathcal{N}(\alpha_t, a^2 I)$.

As in LDA, we now draw a topic z and its words $w_{t,n,d}$ each from a multinomial distribution. While this remains the same in DTM as it was in LDA, we have to keep in mind that the multinomial distribution has a support that must sum up to 1, hence Blei and Lafferty use a parametrization function $\pi()$, that maps β onto its mean: $\pi(\beta_{t,k}) = \frac{\exp(\beta_{t,k})}{\sum_w \exp(\beta_{\beta_{t,k,w}}})$ and equally for θ.

Finally we arrive at a chain of topic models, since each time sequence derives its own topic model.

4.6.2 Inference

Inference of the DTM proves to be more challenging than inference of LDA. This roots from the fact that the normal distribution and the multinomial distribution are no longer conjugate, which makes the posterior difficult to calculate.
While static topic models, such as LDA, can benefit from Gibbs sampling methods, especially collapsed Gibbs sampling, their application is much more challenging in this case (Blei and Lafferty, 2006). Instead the posterior can be approximated using *Variational methods* and in particular *Variational Kalman filters* (presented in section 3.7.2)

5 Applications

5.1 Working on the English wikipedia

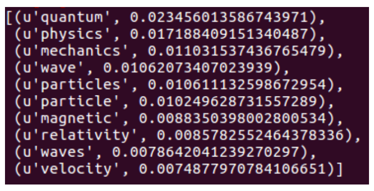
For some of the following examples, I use the English Wikipedia, which can be accessed as an xml version[6]. Using the Python extension *gensim* (Řehůřek and Sojka, 2010) I preprocess the data. Due to the magnitude of size that the English Wikipedia holds - 4.1 million articles are processed - preprocessing takes roughly 12 hours of calculation time on my computer [7], resulting in a 10GB matrix representation and a vocabulary that is limited to 100 000 distinct terms.

5.2 LDA trained on Wikipedia

In this particular instance I used the following model parameters to train the model: 100 topics, symmetric hyper-parameters $\alpha = 0.01$ and $\eta = 0.01$.
For a high degree of precision I decided to update the model every 1000 documents in a total of 3 passes over the whole corpus. Training the model took roughly 10 hours on my machine. The resulting topic distributions seem coherent.

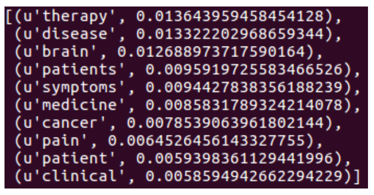
[6]accessed through https://dumps.wikimedia.org/enwiki/

[7]I use a Thinkpad X1 Carbon (2nd gen) with Intel Core i5-4300U and 8GB RAM on Ubuntu 16.04.



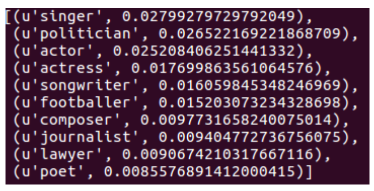
```
[(u'quantum', 0.023456013586743971),
 (u'physics', 0.017188409151340487),
 (u'mechanics', 0.011031537436765479),
 (u'wave', 0.01062073407023939),
 (u'particles', 0.010611132598672954),
 (u'particle', 0.010249628731557289),
 (u'magnetic', 0.0088350398002800534),
 (u'relativity', 0.0085782552464378336),
 (u'waves', 0.0078642041239270297),
 (u'velocity', 0.0074877970784106651)]
```

(a)



```
[(u'therapy', 0.013643959458454128),
 (u'disease', 0.013322202968659344),
 (u'brain', 0.012688973717590164),
 (u'patients', 0.0095919725583466526),
 (u'symptoms', 0.0094427838356188239),
 (u'medicine', 0.0085831789324214078),
 (u'cancer', 0.0078539063961802144),
 (u'pain', 0.0064526456143327755),
 (u'patient', 0.0059398361129441996),
 (u'clinical', 0.0058594942662294229)]
```

(b)



```
[(u'singer', 0.02799279729792049),
 (u'politician', 0.026522169221868709),
 (u'actor', 0.025208406251441332),
 (u'actress', 0.017699863561064576),
 (u'songwriter', 0.016059845348246969),
 (u'footballer', 0.015203073234328698),
 (u'composer', 0.0097731658240075014),
 (u'journalist', 0.009404772736756075),
 (u'lawyer', 0.0090674210317667116),
 (u'poet', 0.0085576891412000415)]
```

(c)

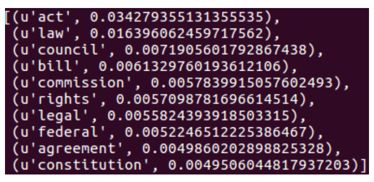
Figure 6: 3 out of 100 topics of a trained LDA model on the English Wikipedia

While LDA is unsupervised and hence does not operate by label, we can easily identify the topics inferred, here for example ”physics-related” (6a), ”medicine-related” (6b) and the third one being a meta topic, which contains professions (6c).

5.2.1 Querying documents

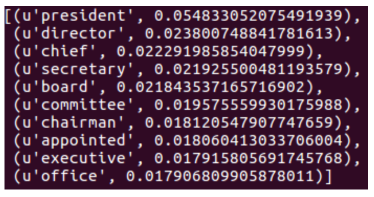
While a trained LDA model on Wikipedia already brings a lot of understanding about the knowledge stored in this massive corpus, LDA becomes very useful when querying new documents (unseen data).
I wrote a script based on Python with several extensions [*gensimLDA*] that reads in a new document, preprocesses it and determines the probability distribution over the topics of the trained model.

For a first example of querying a document, I present hereafter an excerpt of the topic distributions of LDA for the *United States Constitution* as a query document.



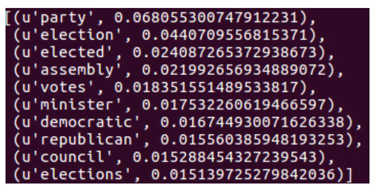
```
[(u'act', 0.034279355131355535),
 (u'law', 0.016396062459717562),
 (u'council', 0.0071905601792867438),
 (u'bill', 0.0061329760193612106),
 (u'commission', 0.0057839915057602493),
 (u'rights', 0.0057098781696614514),
 (u'legal', 0.0055824393918503315),
 (u'federal', 0.0052246512225386467),
 (u'agreement', 0.0049860202898825328),
 (u'constitution', 0.0049506044817937203)]
```

(a) 67.2 %



```
[(u'president', 0.054833052075491939),
 (u'director', 0.023800748841781613),
 (u'chief', 0.022291985854047999),
 (u'secretary', 0.021925500481193579),
 (u'board', 0.021843537165716902),
 (u'committee', 0.019575559930175988),
 (u'chairman', 0.018120547907747659),
 (u'appointed', 0.018060413033706004),
 (u'executive', 0.017915805691745768),
 (u'office', 0.017906809905878011)]
```

(b) 9.4%



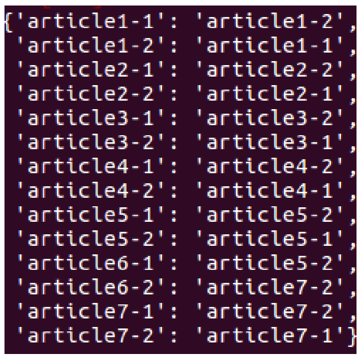
```
[(u'party', 0.068055300747912231),
 (u'election', 0.0440709556815371),
 (u'elected', 0.024087265372938673),
 (u'assembly', 0.021992656934889072),
 (u'votes', 0.018351551489533817),
 (u'minister', 0.017532260619466597),
 (u'democratic', 0.016744930071626338),
 (u'republican', 0.015560385948193253),
 (u'council', 0.015288454327239543),
 (u'elections', 0.015139725279842036)]
```

(c) 6.9%

Figure 7: United States Constitution: 3 topics with the highest probabilities

5.2.2 Matching articles

To test LDA's abilities in a more elaborate way, I create the following task:
I want to let LDA match news articles based on its content. Naturally, the approval of the results would be strongly subjective, such that a rational verification of the script's capabilities would not be possible. In consequence, I take arbitrary news articles from online news sites about different subjects. I proceed by cutting each article into two similar sized pieces, remove all punctuation marks and save them into two different files.
I assume that the topic distribution stays constant throughout each article. A human, understanding what a piece of an article is about, would be able to match the two separated pieces through this knowledge. I query all the pieces of articles as documents. I represent the query results as vectors in the 100-dimensional[8] topic space.
To measure the distance between the different documents, or rather their probability distributions I cannot use *Kullback-Leibler divergence* (see Section 3.6), since the vectors contain zero entries, which make KL divergence inapplicable. Instead I use a simple cosine similarity as presented in section 2.2.10.
I label articles as follows[9]: 'article***i-j***' for the jth ($j \in \{1, 2\}$) part of the ith article
We would expect a result that matches for each i 'article***i*-1**' with 'article***i*-2**'.
In an initial run, using 7 randomly selected articles from different sections of an American news website, the script returns:



```
{'article1-1': 'article1-2',
 'article1-2': 'article1-1',
 'article2-1': 'article2-2',
 'article2-2': 'article2-1',
 'article3-1': 'article3-2',
 'article3-2': 'article3-1',
 'article4-1': 'article4-2',
 'article4-2': 'article4-1',
 'article5-1': 'article5-2',
 'article5-2': 'article5-1',
 'article6-1': 'article5-2',
 'article6-2': 'article7-2',
 'article7-1': 'article7-2',
 'article7-2': 'article7-1'}
```

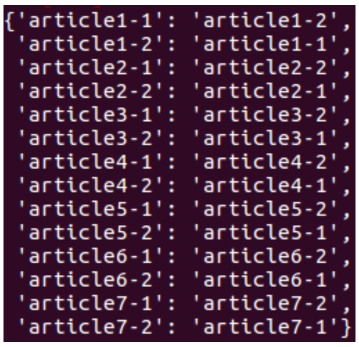
Figure 8: Matching of 14 documents

which is very close to the expected result, except for Article 6.
Why did Article 6 have issues in matching?

[8]For the applications in this section I use a LDA model that was trained on 100 topics.

[9]the naming does not matter to Python since it is not processed, this notation is only for verification purposes afterwords

Looking into the probability distributions of Article 6, I discovered that Article 6 has a very strong distribution of 25.5% over a topic that contains almost exclusively stop word-like terms ('said', 'we', 'went', 'again', 'moved', 'having', etc) that were not filtered as such through the stop word list. Since the information content of such words is very low, it prohibits an accurate document match. As a consequence I created a filter that would remove this topic in the process of matching, leading to the desired result:



```
{'article1-1': 'article1-2',
 'article1-2': 'article1-1',
 'article2-1': 'article2-2',
 'article2-2': 'article2-1',
 'article3-1': 'article3-2',
 'article3-2': 'article3-1',
 'article4-1': 'article4-2',
 'article4-2': 'article4-1',
 'article5-1': 'article5-2',
 'article5-2': 'article5-1',
 'article6-1': 'article6-2',
 'article6-2': 'article6-1',
 'article7-1': 'article7-2',
 'article7-2': 'article7-1'}
```

Figure 9: Matching of 14 documents with filtered topics

After having excluded this topic, the script [*gensimLDA*] works accurately with a higher number of articles as well.

5.3 Dynamic Topic Models in Python

To the best of my knowledge, there is currently no stable and efficient solution in Python for Dynamic Topic models. However, David M. Blei provides the C code for his original paper from 2006[10]. After compiling the C code, I wrote a script based on Python gensim's DTM wrapper that uses Blei's code.

I want to use DTM to show how political events evolve in the media. For this I draw samples of news articles from a major German newspaper, accessible through their web archive. To be precise: For each month, from January 2016 to May 2016 I save between 10 to 15 articles to file, using a bash script to transform to ASCII and to remove all sorts of punctuation [*Doc2ASCII_no_punct*], and preprocess them in the same manner as I preprocessed the LDA queries.

I train the DTM on 5 topics and sort the documents into the 5 time slices. A selection of words found in one of the topics is presented with its evolution over the 5 time slices.

[10]accessible through https://github.com/blei-lab/dtm

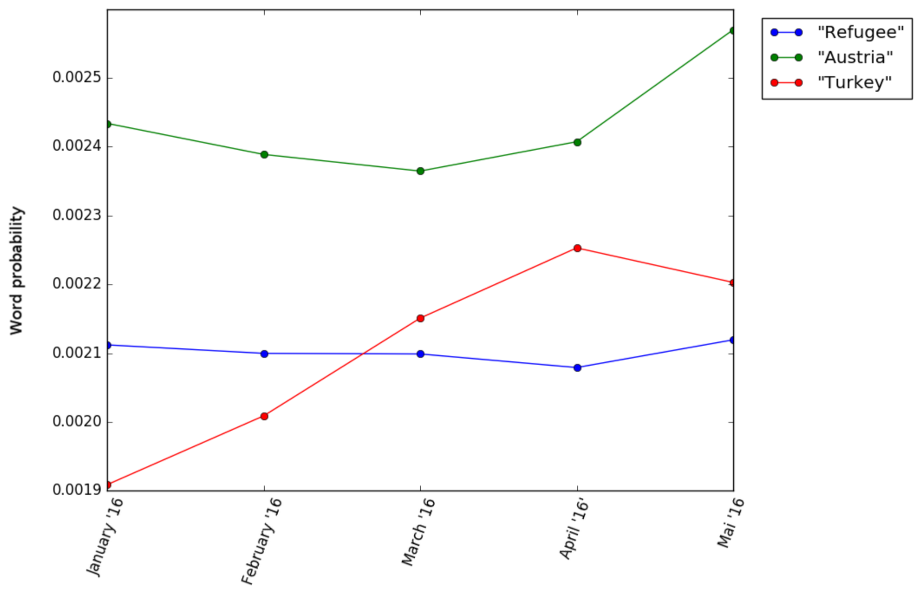


Figure 10: Topic evolution, represented by the three tokens "Refugee", "Austria" and "Turkey". Time range: January 2016 to May 2016. 60 articles in total. [*gensimLDA*]

Due to the time costly process of acquiring the raw documents and the focus of this work, my analysis is limited to 5 months. The results, however, show that an extension to a magnitude of several years could lead to valuable insights into media understanding.

6 Conclusions

Topic models are powerful tools for understanding latent structures and especially the latent topics of documents. In the previous sections, I introduced numerous concepts crucial to the understanding of topic models. Some of these concepts are of pure statistical origin, some have a background in information theory or even linguistics.
Using LDA, trained on the English Wikipedia on 100 topics, I built an example of how LDA can be used to understand query documents. I introduce document querying by using the United States Constitution to show how this trained model understands the content, particularly how the US Constitution is distributed over the topics. I then introduce a script that successfully matches news articles on the basis of its content.

Benefiting from the time dimension of Dynamic Topic Models, I introduce an application in the field of media coverage of political events, where I analyse German news articles with respect to their publishing months to show the evolution of a topic concerning terms related to Germany's refugee crisis.

I introduced topic models almost entirely from the perspective of natural language processing as it is their native field. After all, a document of text is easily replaceable with another concept; for this I would like to point out a few examples, which without doubt only represent a small fraction of the fields where topic models can find application.
A large and growing field in which topic models are to be found is computer vision, especially image classification and meta-data generation of images. Chong et al. (2009) use a supervised (labeled) extension of the LDA, called *sLDA* to annotate images, finding describing tags and then classify them into categories.
A very recent field where topic models are applied to is *bioinformatics*. Liu et al. (2016) explains this by the good interpretability of output created by topic models and a strongly growing amount of biological data. Many studies use topic models to understand large genomic data sets, where gene samples are handled like documents, assuming latent topics to generate the gene samples Liu et al. (2016).

Loosening or tightening certain restrictions of the *Latent Dirichlet Allocation* assumptions allow for a strong adaptability to new environments and cases of application. The interpretability and human-like intuition behind the model make it a strong tool in modern machine learning.

References

Airoldi, E. M., D. Blei, E. A. Erosheva, and S. E. Fienberg (2014): *Handbook of Mixed Membership Models and Their Applications*, Chapman & Hall/CRC, 1st ed.

Baeza-Yates, R. A. and B. Ribeiro-Neto (1999): *Modern Information Retrieval*, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2 ed.

Blei, D. M. (2012): "Probabilistic topic models," *Communications of the ACM*, 55, 77–84.

Blei, D. M. and J. D. Lafferty (2006): "Dynamic Topic Models," in *Proceedings of the 23rd International Conference on Machine Learning*, New York, NY, USA: ACM, ICML '06, 113–120.

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

Bush, V. (1945): "As we may think," *Atlantic Monthly*, 176, 101–108.

Chen, M.-H., Q.-M. Shao, and J. G. Ibrahim (2000): *Monte Carlo Methods in Bayesian Computation*, Springer-Verlag New York, 1st ed.

Chong, W., D. Blei, and F.-F. Li (2009): "Simultaneous image classification and annotation," *Computer Vision and Pattern Recognition.*

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman (1990): "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, 41, 391–407.

Everitt, B. S. (1984): "Introduction to Latent Variable Models," *Biometrical Journal.*

Fox, C. (1992): *Lexical Analysis and Stoplist*, W. Frakes and R. Baeza-Yates, chap. 7, 102–130.

Härdle, W. K. and L. Simar (2012): *Applied Multivariate Statistical Analysis*, Berlin, Germany: Springer-Verlag, 3 ed.

Hiemstra, D. (2000): "A probabilistic justification for using tfidf term weighting in information retrieval," *International Journal on Digital Libraries*, 3, 131–139.

Hofmann, T. (1999): "Probabilistic Latent Semantic Analysis," in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., UAI'99, 289–296.

——— (2001): "Unsupervised Learning by Probabilistic Latent Semantic Analysis," *Mach. Learn.*, 42, 177–196.

Jeffrey, A. (2004): "11 - The Gamma, Beta, Pi, and Psi Functions," in *Handbook of Mathematical Formulas and Integrals (Third Edition)*, ed. by A. Jeffrey, Burlington: Academic Press, 221 – 228, third edition ed.

Lauritzen, S. (2007): "Exchangeability and de Finettis Theorems," .

Liu, L., L. Tang, W. Dong, S. Yao, and W. Zhou (2016): "An overview of topic modeling and its current applications in bioinformatics," *SpringerPlus*, 5, 1608.

Luhn, H. P. (1957): "A Statistical Approach to Mechanized Encoding and Searching of Literary Information," *IBM J. Res. Dev.*, 1, 309–317.

Maybeck, P. S. (1979): *Stochastic models, estimation, and control*, vol. 141 of *Mathematics in Science and Engineering.*

Murphy, K. P. (2012): *Machine Learning: A Probabilistic Perspective*, The MIT Press.

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

Salton, G., A. Wong, and C. S. Yang (1975): "A Vector Space Model for Automatic Indexing," *Commun. ACM*, 18, 613–620.

Saul, L. and M. I. Jordan (1995): "Exploiting Tractable Substructures in Intractable Networks," in *Advances in Neural Information Processing Systems 8*, MIT Press, 486–492.

Wasserman, L. (2010): *All of Statistics: A Concise Course in Statistical Inference*, Springer Publishing Company, Incorporated.

Declaration of Authorship

I, Jérôme Bau, hereby declare that I have not previously submitted the present work for other examinations. I wrote this work independently. All sources, including sources from the Internet, that I have reproduced in either an unaltered or modified form (particularly sources for texts, graphs, tables and images), have been acknowledged by me as such. I understand that violations of these principles will result in proceedings regarding deception or attempted deception.

Paris, November 22, 2016

(Jérôme Bau)