UCLouvain

FACULTE DES SCIENCES

Unsupervised topic modeling for short documents

Author: **Oscar Liessens**
Advisor: **Eugen Pircalabelu**
Academic year 2020-2021
Master in data science, statistics orientation

Unsupervised topic modeling for short documents

Author
Oscar Liessens

Reader
Christian Ritter

Advisor
Eugen Pircalabelu

Reader
Séverine Guisset

Academic year 2020-2021
Master in data science, statistics orientation

Louvain School of Statistics,
Biostatistics and Actuarial Sciences

Acknowledgments

I obviously dedicate this thesis to the memory of my father, the kindest, toughest, funniest, wisest man I ever had the chance to meet. I owe him so much of everything I do, everything I am.

I am particularly grateful to my mom who kept believing in me beyond all reason, in the most loving and pressure-free way there was. I am grateful to Félix, my little brother who somehow manages to be both the devil and the angel I need on my shoulder. I am grateful to Gaspard, my sweet little brother whose insane resilience in the face of everything life threw at him keeps inspiring me to be braver. I am grateful to everyone else, to my cherished close family, to my dear friends, to every well-meaning stranger who asked me even once how this thesis was going.

Finally, I am immensely grateful to Eugen, my advisor. Anybody who knows me will also know that I had a hard time finishing this thesis and that I made his task quite difficult. But his unwavering support and his patient kindness kept me going. I will long remember his masterful way to make me come up with answers to my own questions, which helped me learn a lot more than I would normally have. Thank you, Eugen, for allowing me to finish this work and be proud of it. This means everything to me.

Contents

Introduction

The world can, in a purely scientific way, be constructed as an intractably large amount of signals carrying information about matter. Among those signals, some can be perceived and processed by humans. It is the first work of conscious thinking to organize these stimuli into coherent semantic units. Thus, it should come as no surprise that this is also a most important enterprise in modern machine learning and artificial intelligence research.

Topic modeling is the action of modeling high-dimensional data by assuming the existence of underlying abstract topics. One example of this kind of data is, of course, natural language. In this context, topic modeling means to obtain a generative model of speech wherein the occurrence of a given word in a given unit of language is associated with the occurrence of an abstract, unobserved topic. It is also often assumed that each unit of speech exhibits more than one underlying topic, in different amounts. Therefore, each topic can be characterized by both the words with which it is likely to be associated and the documents in which it is likely to be present. If topic modeling is most easily understood and often explained exclusively in terms of natural language processing, its usefulness does not stop there. The essential idea is simply to summarize the occurrence of a multitude of signals in a multitude of observations by modeling underlying trends. The search for meaningful semantic units in vast arrays of features is ubiquitous, from fields like machine vision to the processing of genomics data. More than mimicking the human mind, the act of finding hidden topics in high-dimensional data goes beyond its present capabilities.

It is common knowledge that the amount of data humans generate keeps growing exponentially. It is also often suggested that most of these data are available in an unstructured format, largely unavailable to common quantitative analysis but ready for consumption by humans. Unfortunately, our natural tendency towards laziness and the sheer quantity of the information at our disposal make it very difficult to organize this into actionable knowledge. Most of the time, this results in a generalized preference for bite-sized internet content. This is an era-defining tendency, which can be seen in the rise of social networks and micro-blogging, in the re-orientation of otherwise reputable media towards click-baiting headlines, and in the expansion of our expression modes towards other channels such as memes. Ceaselessly prompted to express themselves and to stay in constant communication with the world, humans will also express themselves in a prompt fashion. The text data we generate this way are indeed invaluable, whether these are used for marketing

purposes, to inform political campaigns or to find deep insight into public opinion at key moments of our history. Nevertheless, this text frequently presents a characteristic that is often disregarded in the natural language processing field; it's short.

This simple fact raises complex issues for topic modeling, which can be addressed with different creative solutions. The objective of this thesis is to compare the merits and difficulties of these methods, both on the theoretical plan and using real user-generated data. To lay the necessary foundations, we will begin by going over various text mining conventions and detailing some notations we will be using going forward. We will then go over different topic modeling such as Probabilistic Latent Semantic Analysis (pLSA; Hofmann, 1999), Latent Dirichlet Allocation (LDA; Blei et al., 2003) and Non-negative Matrix Factorization (NMF; Lee and Seung, 1999). Using mainly LDA as a basis, we will then present different ways in which topic modeling can be improved to address the high sparsity of short documents. These techniques include grouping short texts into longer ones (Quan et al., 2015; Zuo et al., 2016) and modeling directly the co-occurences of the words (Yan, Guo, Lan, et al., 2013; Yan, Guo, Liu, et al., 2013). Finally, we will wrap the theoretical part by reviewing different techniques that can be used to evaluate the quality of a topic model. In the second part of the thesis, we will compare different topic modeling schemes on simulated data and real life-world short documents, using carefully selected evaluation methods. The results of these experiments will then be discussed at length, with special attention paid to human interpretability of the yielded topics.

Part I

Literature review

Chapter 1

Basics in text mining

The most common form in which text data present themselves is called a corpus, i.e. a collection of documents. Documents are semantic units composed of words arranged in a certain order. A fundamental problem in mining these text data for information is the need to translate written language, intelligible to humans only, to quantities that can be understood by machines as well. This implies the need to filter out words that are helpful for human understanding, but not for machines. One of the first fields to study these issues was that of information retrieval (IR; Baeza-Yates and Ribeiro, 2011), i.e. the art of querying a storage system and receiving the most relevant information.

1.1 Notation conventions

In order to proceed in an orderly fashion, it is important to define precisely some notations and terminology that will be used throughout the present work.

Vocabulary. The vocabulary is the set of all possible words, i.e. all the words encountered in the corpus. The set has size V.

Word. The "word", or token, is the basic unit of our language understanding paradigm. It is worth mentioning that the chosen units can be unigrams, i.e. single isolated words, or n-grams, i.e. tokens consisting of n consequent words. Nevertheless, we will focus on unigrams and refer to them as words for the remainder of this work. In our paradigm, we will denote words with unit vectors w of size V such that the v^{th} word of the vocabulary has 1 as its v^{th} value and 0 elsewhere.

Document. Documents are series of N words, and are therefore represented as matrices of size $N \times V$. The n^{th} word of a document will be denoted by w_n. Nevertheless, in general practice and under the bag-of-words (bow; see Section 1.3) assumption, we will prefer representing them as denser vectors of size V rather than matrices. The process of aggregating the sparse values of documents to dense vectors is described further down, in Section 1.3. Documents are denoted as D in their matrix form,

and d in their vector forms. Additionally, a subscript may be added to denote the representation framework of a vector document, such as d_c for a document in the count representation.

Corpus. The corpus is a set of M documents, denoted by $\mathcal{C}$, where the m^{th} document is denoted by D_m (or d_m if the document is in a vectorized representation). Each of these documents has N_m elements. If the documents are vectorized, as described in Section 1.3, the corpus can be seen as a matrix C having the M documents as rows and the V words as columns. In this case, a subscript may be added as index to denote the representation framework of the documents when relevant, such as C_c for a corpus in the count representation. We will also denote the number of occurrences of the v^{th} word of the vocabulary in the m^{th} document by C_{mv}. To denote a vector containing either all the words for a document or all the documents for a word, we will respectively use $C_{m\bullet}$ and $C_{\bullet v}$.

Finally, let us also clarify the following mathematical notation conventions:

- The transposed of a matrix A will be denoted A^T;
- The L_1 norm of a vector a will be denoted $\|a\|_1$;
- The probability of an event E will be denoted $P(E)$;
- For a matrix A, the element at the intersection of the i^{th} row and the j^{th} column will be denoted by A_{ij};
- The indicator function, worth 1 in case the event E is true and 0 in any other case, is denoted $\mathbf{1}_{\{E\}}$.

For future references, all mathematical notations used in this thesis can be found in Appendix A1. The same goes for the acronyms, which can all be found in Appendix A2.

1.2 Necessary preprocessing of text data

The human language, in all its instances, is littered with non-informative words and variations of the same semantic units. Therefore, in most text mining applications arises a need for a thorough data preprocessing. Fine-tuning and domains-specific tweaking are the norm in this area, but some steps are highly usual and recommendable. The fine-tuning of this process is of crucial importance, as the decisions that are taken here will certainly be reflected in the models that can eventually be computed.

Tokenizing. In the context of text mining, the act of tokenizing consists in converting documents that are encoded as one long string to lists of units. These units can be words, bigrams (i.e. combinations of two words), n-grams (i.e. combinations of n

words) or even sentences. This is a crucial step, as the counts of these tokens will later be used in order to compute models of the corpus. In topic modeling, word tokens are most frequently used.

Uppercase and punctuation removal. To computers, the difference between lower and uppercase matters; it is therefore good practice to put all documents in lowercase only. The punctuation, that carries no significant meaning when the analysis is done on a word basis without regard for their order, can be systematically removed as well.

> *Example: This thesis is about topic modeling; it is a subject that, arguably, interests artificial intelligence researchers from all fields.*
> *→ this thesis is about topic modeling it is a subject that arguably interests artificial intelligence researchers from all fields*

Stopwords removal. Stopwords are all words that are considered both frequent and non-informative for a specific task. There exist, for each language, corpora of the usual stopwords that allow the user to remove them at once. Additionally, in the context of topic modeling, all words that are expected to be in all documents can be added to the list of stopwords. This can be done based on the frequency of words across documents.

> *Example: this thesis is about topic modeling it is a subject that arguably interests artificial intelligence researchers from all fields*
> *→ thesis topic modeling subject arguably interests artificial intelligence researchers fields*

Stemming. In natural language, a word can have multiple variations on its suffix that essentially carry the same meaning. In order to group all these variations under a single token summarizing the information, it is common practice in text mining to cut the suffixes of all words to the common root they share with other semantically similar words.

> *Example: thesis topic modeling subject arguably interests artificial intelligence researchers fields*
> *→ thesi topic model subject arguabl interest artifici intellig research field*

Lemmatizing. This is essentially the same as stemming, but with a special care toward conserving words existing in the language. With lemmatizing, all semantically related variations of a word are grouped under the banner of a single, simpler word.

> *Example: thesis topic modeling subject arguably interests artificial intelligence researchers fields*
> *→ thesis topic modeling subject arguably interest artificial intelligence researcher field*

1.3 Vector representations of documents

Most methods to represent documents D as dense vectors d make the assumption that the order of the words does not matter, which is known as the bag-of-words (bow) assumption. This is of course not true, and there exist multiple extensions to relax this assumption which are beyond the object of this thesis. However, this assumption is a very frequent one, and it allows to have a simple working representation of documents.

Count representation In its most basic aspect, a document is a matrix D which has N word vectors as its rows. As each word can be modeled as a random vector following a Multinomial distribution with only one trial, it follows that, if we forgo the order of the words, the whole document can be modeled as a random vector following a Multinomial distribution with N trials. In this paradigm, which we will further call the *count representation*, the resulting d vector is simply the sum of the columns of D. Therefore, with 1_N a column vector of ones having size N, the count representation of a document is given by:

$$d_c = D^T \times 1_N \quad .$$

For instance, let us assume we have the two following sentences:

- *Sentence 1: This thesis is about topic modeling; it is a subject that, arguably, interests artificial intelligence researchers from all fields.*
- *Sentence 2: Theses are long research papers submitted by students at the end of their studies. An example of thesis subject is, for instance, topic modeling.*

Once processed with the method detailed above using lemmatizing, these sentences become:

- *Sentence 1: thesis topic modeling subject arguably interest artificial intelligence researcher field*
- *Sentence 2: thesis long research paper submitted student end study example thesis subject instance topic modeling*

These sentences, vectorized using the count representations, are then as shown in Table 1.1, under the "Count" columns. It is important to note that, for presentation reasons, these data are shown in a transposed way relative to what we discussed earlier. Words, which are here presented as rows, should be columns. In general a vectorized corpus will be wide rather than long, as the vocabulary is oftentimes larger than the number of documents.

	Count		TF		TF-IDF	
	Sentence 1	Sentence 2	Sentence 1	Sentence 2	Sentence 1	Sentence 2
arguably	1	0	0.100	0.000	0.353	0.000
artificial	1	0	0.100	0.000	0.353	0.000
end	0	1	0.000	0.071	0.000	0.282
example	0	1	0.000	0.071	0.000	0.282
field	1	0	0.100	0.000	0.353	0.000
instance	0	1	0.000	0.071	0.000	0.282
intelligence	1	0	0.100	0.000	0.353	0.000
interest	1	0	0.100	0.000	0.353	0.000
long	0	1	0.000	0.071	0.000	0.282
modeling	1	1	0.100	0.071	0.251	0.201
paper	0	1	0.000	0.071	0.000	0.282
research	0	1	0.000	0.071	0.000	0.282
researcher	1	0	0.100	0.000	0.353	0.000
student	0	1	0.000	0.071	0.000	0.282
study	0	1	0.000	0.071	0.000	0.282
subject	1	1	0.100	0.071	0.251	0.201
submitted	0	1	0.000	0.071	0.000	0.282
thesis	1	2	0.100	0.143	0.251	0.402
topic	1	1	0.100	0.071	0.251	0.201

Table 1.1: Example of count, TF and TF-IDF vectorization of two sentences

TF representation One problem with the count vectorization is that some words will be labeled much more present in large documents that simply contain more words in general. A more advanced and still tractable solution is to measure the occurrences of each word and to weigh it against the size of the document, therefore obtaining frequencies that can be compared across documents of different sizes. In IR, this is known as the TF (term frequency) representation:

$$d_{tf} = \frac{d_c}{N} \quad .$$

For instance, using the two same sentences as above, we get the representations shown under the "TF" columns of Table 1.1. We can see that words appearing only once in both sentences have more weight in the first one, as it is shorter.

TF-IDF representation Unfortunately, this still means that some words that appear indiscriminately in all documents will carry the most weight. In the TF-IDF (term frequency-inverse document frequency) representation, the frequency of each word in the documents is weighted with its frequency in the entire corpus. Let us then denote by w_{df} a row vector

of size V having for values the total number of occurrences for each of the V words. For instance, with the number of documents in the corpus being M, for a corpus C, that is represented as a matrix of counts, w_{df} is given by $\sum_{m=1}^{M} C_{m\bullet}$. Let us also clarify that we here consider the logarithm of a vector to simply be the vector comprised of the logarithm of each of its elements. The TF-IDF representation of a document is then given by:

$$d_{tfidf} = \frac{d_{tf}}{\log(\frac{1+w_{df}}{1+M}) + 1} \quad .$$

This scheme will give less weight to words that are widely present across the documents, which one can see as a continuous alternative to the removal of discrete stopwords. Again using the two sentences shown above, using TF-IDF, we get the representations shown under the "TF-IDF" columns of Table 1.1. It can be seen that, in addition to the same word occurring in documents of different length having different weights, words that are only present in one of the two sentences have a higher weight.

This, in a way, already constitutes a model of natural language and a kind of primitive topic modeling. But, in this instance, the information is not very synthesized and the extracted topics contain each only one word.

1.4 Probabilistic Latent Semantic Analysis

To gain more insight into the documents structure, researchers in IR have proposed another way of modeling language known as Latent Semantic Analysis (LSA; Deerwester et al., 1990). In this paradigm, one performs a singular value decomposition (SVD) on the count representations of documents to obtain features summarizing the distribution of words in a lower dimensional space. In a more practical way, let us assume we have M count representations of documents and a vocabulary of size V, therefore giving a corpus matrix C of dimensions $M \times V$. Using only rudimentary linear algebra, one can decompose C as a product of Σ, a diagonal matrix containing the singular values, and two matrices U_1 and U_2 containing, respectively, the associated row and column singular vectors.

By setting all singular values in Σ to zero but the K largest, one obtains next a $\tilde{\Sigma}_K$ matrix that, when used in (1.1), will allow to approximate the original matrix C. This leaves us with what can be interpreted as K latent semantic topics. In the lower dimensional space, the coordinates of the documents are simply given by $U_1 \times \tilde{\Sigma}_K$ and those of the words by $\tilde{\Sigma}_K \times U_2^T$. These coordinates can be interpreted as quantifying the association of both documents and words with the latent topics, thus defining them. This corresponds, in essence, to performing a principal component analysis on a matrix where columns are words and rows are documents.

$$C = U_1 \times \Sigma \times U_2^T \tag{1.1}$$
$$C \approx U_1 \times \tilde{\Sigma}_K \times U_2^T \tag{1.2}$$

LSA possesses many attributes of modern topic modeling, but it also has some shortcomings, not the least being that it lacks a simple probabilistic interpretation. A first step towards changing that was taken when Probabilistic Latent Semantic Analysis (pLSA; Hofmann, 1999) was proposed. More than an algebraic approximation of the original matrix representation of the corpus, this method relies heavily on previous work about statistical modeling of latent features like the aspect model (Hofmann et al., 1998) and suggests a first generative model of natural language. In this context, we define a latent, unobserved factor z that summarizes the association between the documents d and the words w, and against which the later are assumed conditionally independent of each other. We consider the latent variable z to have K levels. It is important to note that here topics, words and documents are denoted by dummy variables and are therefore unit vectors of respective sizes K, V and N. The probability of a word and a document appearing together can therefore be formalized as follows (Hofmann, 1999).

$$\begin{aligned} P(d, w) &= P(d)P(w \mid d) \\ P(w \mid d) &= \sum_{k=1}^{K} P(w \mid z_k)P(z_k \mid d) \\ P(d, w) &= \sum_{k=1}^{K} P(w \mid z_k)P(z_k \mid d)P(d) \\ &= \sum_{k=1}^{K} P(w \mid z_k)P(z_k, d) \\ &= \sum_{k=1}^{K} P(z_k)P(d \mid z_k)P(w \mid z_k) \end{aligned} \tag{1.3}$$

The dimensionality of z, i.e. the number of topics K, is less than the number of documents M or the vocabulary size V. Therefore, knowing both the conditional distribution of the topics given the document, i.e. $P(z_k \mid d)$, and the conditional distribution of the words given the topics, i.e. $P(w \mid z_k)$, means having an approximation of their original distributions in a lower dimensional space.

In order to estimate the optimal parameters of the aforementioned multinomial distributions, Hofmann (1999) suggested using the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). The steps are then as follows:

1. E-step: The expected probabilities for the multinomial variable z, conditionally on the word w and the document d, are computed using the Bayes formula (see equation 2.1). It is worth mentioning that, the first time this step is executed, one does not yet have any estimate for these probabilities and arbitrary values can be used. We thus get the following equation to compute the E-step:

$$P(z \mid d, w) = \frac{P(z)P(d \mid z)P(w \mid z)}{\sum_{k=1}^{K} P(z_k)P(d \mid z_k)P(w \mid z_k)}$$

2. M-step: Using the expectation for the conditional probabilities of z, computed in the E-step, one can update the conditional probabilities of w knowing z and d knowing z, and the marginal probabilities of z. Using the formulas above (see equation 1.3) as well as the results of the E-step, we thus have the following equations for the M-step:

$$P(w \mid z) = \sum_{m=1}^{M} C_{m\bullet} P(z \mid d_m)$$
$$P(d \mid z) = \sum_{v=1}^{V} C_{\bullet v} P(z \mid w_v)$$
$$P(z) = \sum_{m=1}^{M} \sum_{v=1}^{V} C_{mv} P(z \mid d_m, w_v)$$

Repeating these steps until convergence is achieved allows one to obtain maximum likelihood estimation of the probabilities of an unobserved, "missing" quantity, i.e. the latent variable z.

Using maximum likelihood to estimate the model allows to find the latent variable z that explains best the occurrences of words and documents, instead of just trying to minimize the error of the approximation (i.e. equation 1.2). Moreover, it offers a probabilistic interpretation of the process and allows using classical statistics to find an optimal number of topics (Hofmann, 1999).

This model of natural language, while an important stepping stone to later models, suffers from certain shortcomings. As stated by Blei et al. (2003), d is considered a multinomial variable with as many categories as there are documents in the corpus. Therefore, the model only contains information about those documents and, by essence, lacks a generalizing capability. There is no meaningful way in which a learned model can be used to assign probabilities of topics to new documents, or in which it can be used as a generative model. Moreover, the multinomial probabilities of topics appearing in documents $P(z \mid d)$

are parameters to be estimated with the EM algorithm. This means that the number of parameters to be estimated is, with K the number of topics, V the size of the vocabulary and M the size of the corpus, $K \times V + K \times M$, and therefore grows linearly with the size of the corpus (Blei et al., 2003). While in other contexts this complexity would be considered desirable, it offers too little reduction in description length when it comes to natural language. Indeed, the amount of available text data, regardless of context, has a way of growing exponentially, which can quickly make this model impractical. One way to reduce the number of parameters to estimate would be to describe the prior probabilities of the document mixtures of topics instead of the mixtures themselves, for instance.

Chapter 2

Latent Dirichlet Allocation

The Latent Dirichlet Allocation (LDA), proposed by Blei et al. (2003), remedies the shortcomings of pLSA and pushes the modeling of natural language one step forward. By assigning a probability distribution to almost every aspect of the language generative process, the LDA reduces drastically the number of estimated parameters and allows for new computation methods and extensions. The LDA is the basis of most modern topic modeling research, to the extent that some authors seem to find it synonymous to topic modeling (Asuncion et al., 2009; Newman, Karimi, et al., 2009). Countless extensions, relaxing various assumptions of the paradigm (Blei & Lafferty, 2006, 2007; Doyle & Elkan, 2009; Teh et al., 2006), defining new estimation methods (Asuncion et al., 2009; Hoffman et al., 2010; C. Wang & Blei, 2009) and finding applications in different contexts (Chen et al., 2010; Erosheva et al., 2004; Fei-Fei & Perona, 2005; Rosen-Zvi et al., 2004), have been proposed.

2.1 Introduction to the Bayesian framework

The LDA is a hierarchical Bayesian model, and as such some notions of the Bayesian framework can be helpful to understand the terminology we will use. At their core, Bayesian statistics rely on the notion that probabilities are quantifications of reasonable expectations about the occurrence of an event and should be modeled as such. In this interpretation, the theorem of Bayes becomes an important tool for representing the scientific process.

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)} \propto P(B \mid A) \times P(A) \tag{2.1}$$

$$Posterior \propto Likelihood \times Prior \tag{2.2}$$

The prior is a probability distribution that should represent the best current knowledge or belief about the parameters of the model. The likelihood, on the other hand, is a function of the model parameters yielded by the joint probability distribution of the data. It is often regarded as a function of the parameters only, the data being assumed fixed.

The Bayes formula stays true whether we consider discrete probabilities or continuous probability distributions. In this last case, the priors and posteriors are often the probabilities of the parameters used in the likelihood function. For each possible value of the priors, we find the posterior probability of this value given the data. In practice, the denominator in the Bayes formula is often omitted because its computation can be challenging, and it is not necessary if we normalize the numerator so that it integrates to 1. We thus obtain a quantity proportional to the posterior probability. The posterior represents the plausibility of the prior being true with regard to the data, and therefore is an excellent formulation of the act of updating one's belief according to new evidence.

In Bayesian practice, some distributions are typically used together. A good example of this are the Beta and Binomial distributions. Let d_v be the number of times a word appears in a document and be such that $d_v \sim Binom(N, \pi)$, with N the number of words in the document and π the probability of the word appearing in it. In a Bayesian framework, the parameter π is considered a random variable itself and will be attributed a prior distribution of its own. Since the Beta distribution is defined on the interval $[0, 1]$, it is often used to represent the prior distribution of π such that $\pi \sim Beta(a, b)$. Here, a and b are shape parameters whose order of magnitude can be used to reflect the level of confidence in this prior knowledge. Furthermore, the posterior resulting from using a Beta prior and a Binomial likelihood follows a Beta distribution, the same as the prior albeit with different parameters. These are called conjugate distributions.

If we extrapolate this situation to higher dimensional spaces, we have the same situation with the Multinomial and Dirichlet distributions. The Multinomial distribution is typically used to represent the results of experiments with multiple possible, mutually exclusive results, while the Dirichlet distribution is an extension of the Beta distribution used to model variables taking multiple values adding up to 1. Let us now denote d, a vector of length V representing a document in vectorized form (as discussed in Section 1.3), with each of its element the number of times each word of the vocabulary has appeared in it. It is such that $d \sim Multinom(N, \pi)$, where $\pi = [\pi_1 \dots \pi_V]^T$ is a vector of size V containing the probability of every word to appear. We can then use the Dirichlet distribution as prior for π such that $\pi \sim Dir(\alpha)$. Here, α is a vector of size V, whose strictly positive values are usually made identical but which can be adjusted to reflect asymmetries in the word probabilities and the level of confidence in the prior.

2.2 Intuition

The LDA (Blei et al., 2003) uses this mechanism in order to represent, in a probabilistically sound manner, every aspect of the paradigm. One of the main advantages of this approach is that it is possible, using only probability distributions, to describe how a corpus would be created using a few base parameters. The authors (Blei et al., 2003) call this the generative process of the LDA.

We have already described some information that one needs to have before creating a corpus. Indeed, we need to decide on a number of documents, M, and a vocabulary of size V. The number of topics, K, will also be assumed known beforehand.

Before the beginning of the process, it is therefore assumed we know (or decided arbitrarily) the following values:

- ζ, the average number of words per document;
- V, the number of possible words;
- M, the number of documents;
- K, the number of topics;
- α, a K-length vector of strictly positive values;
- η, a V-length vector of strictly positive values.

Using these values, we can then sample from several probability distributions to get more information about the sampled corpus:

- Sample M times from $N \sim Pois(\zeta)$, yielding the length of each document, i.e. an M-length vector of integers. We later refer to the length of the m^{th} document as N_m.
- Sample M times from $\theta \sim Dir(\alpha)$, yielding the probability of each topic for each document, or mixing proportions, i.e. a total of K vectors of length M forming an $M \times K$ matrix. We later refer to the mixing proportions of the m^{th} document as θ_m.
- Sample K times from $\beta \sim Dir(\eta)$, yielding the probability of each word for each topic, i.e. a total of K vectors of length V forming a $K \times V$ matrix. We later refer to the words probabilities of the k^{th} topic as β_k.

Finally, having all the information we need, we can proceed to the building of the corpus itself. For each of the M documents indexed by m, for each of its N words indexed by n:

1. Sample once from $z_{mn} \sim Multinom(1, \theta_m)$, yielding a K-length unit vector having 1 as the k^{th} value. This means that w_{wn}, the n^{th} word in the m^{th} document, will be chosen with the specific probabilities of the k^{th} topic.
2. Sample once from $w_{mn} \sim Multinom(1, \beta_k)$, yielding a V-length unit vector having 1 as the v^{th} value. This means that the n^{th} word of the m^{th} document will be the v^{th} word of the vocabulary.

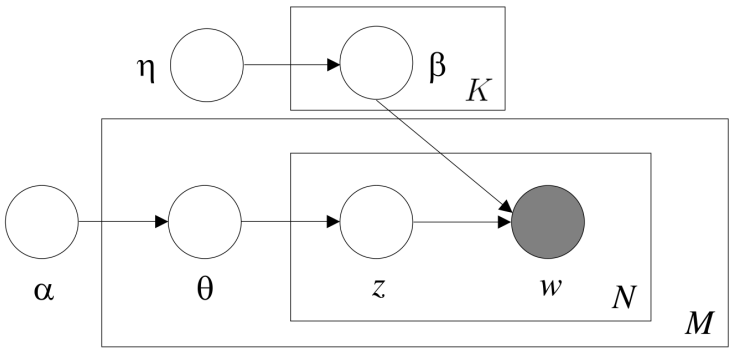

Figure 2.1: Graphical model representation of the smoothed LDA. The outer box represents repetition for each document, the inner box represents repetition for each word, and the upper box represents repetition for each topic (from Blei et al., 2003)

Following these steps, we then have a clearly defined corpus. Because most quantities involved depend on others, viewing the graphical representation of the model may help with understanding (see Fig. 2.1). To obtain a matrix representation of it, one just needs to sum up the word vectors of each document, yielding count-vectorized documents. Using them as rows, one therefore has an $M \times V$ matrix with the word counts as columns. Unfortunately, while this gives a clear intuition of the soundness of the LDA paradigm, the real issue is most often to do the opposite, and go from an existing corpus to its parameters. This is what will be discussed during the next steps.

2.3 Computation problem

From the generative process, we can deduce a joint distribution for both observed and hidden variables (see equation 2.3). In this simple formula, the dependencies described above and shown in Fig. 2.1 are evident as well. The probability with which a given word w_{mn} is chosen depends on the topic z and the β topic-word probabilities. In turn, the choice of z_{mn} depends on the θ_m document-topic probabilities.

$$P(\theta, \beta, z, \mathcal{C}) = \prod_{k=1}^{K}[P(\beta_k)] \prod_{m=1}^{M}[P(\theta_m) \prod_{n=1}^{N} P(z_{mn} \mid \theta_m) P(w_{mn} \mid \beta_k, z_{mn})] \tag{2.3}$$

The goal, in a real use case, is to estimate the distributions of the hidden variables, θ, β and z, from the observed data, i.e. the corpus $\mathcal{C}$. In other words, we want to obtain the posterior distribution, which is the new distribution of the parameters given the corpus. It is a distribution on the same parameter as the prior distributions, but with the added information of the corpus. Using simple probability properties and the Bayes theorem,

discarding the length of the documents which is not useful in computing topic models for a given corpus, we can easily obtain the posterior.

$$\begin{aligned} P(\theta, \beta, z \mid \mathcal{C}) =& \frac{P(\mathcal{C} \mid \theta, \beta, z) P(\theta, \beta, z)}{P(\mathcal{C})} \\ =& \frac{P(\theta, \beta, z, \mathcal{C})}{\iiint P(\theta, \beta, z, \mathcal{C}) \, d\theta \, d\beta \, dz} \end{aligned} \tag{2.4}$$

The numerator of (2.4) is the joint distribution of the hidden variables and the corpus, and can be easily computed for any value of the hidden variables. On the other hand the denominator in (2.4) is intractable to compute, because it would imply integrating θ, β and z out of the joint distribution. This corresponds, in practice, to summing over all possible combinations of word and topic, for each word of each document. This a priori prevents the use of the posterior in a simple EM algorithm as was discussed in Section 1.4, for instance.

2.4 Variational inference

Fortunately, this issue is a frequent one in Bayesian modeling and there exist common solutions. Markov chain Monte Carlo (MCMC) methods, for instance, form a well-known class of algorithms allowing to obtain a random sample of any probability distribution (Griffiths & Steyvers, 2004; Griffiths et al., 2004; Steyvers & Griffiths, 2007). These methods rely on a stochastic process, wherein one constructs a Markov chain with the distribution of interest as equilibrium. This allows, once this state of equilibrium is reached, to obtain a random sample of the distribution. This is a powerful kind of technique, as there exist variations for virtually any situation one can encounter, and it can quickly be implemented for any new use case without too much technical knowledge. Nevertheless, the stochastic process is computationally intensive — particularly so when the goal is to obtain a sample from a high dimensional distribution.

Variational inference (VI) is another class of methods that can help solve the issue of intractable distributions in a more deterministic manner. Both MCMC and variational methods can be and have been used to compute probabilistic topic models, but in the original presentation of the LDA paradigm the proposed algorithm was variational (Blei et al., 2003). Moreover, variational methods have shown performances on par with those of MCMC at greater speeds (Asuncion et al., 2009; Jordan et al., 1999).

Instead of sampling stochastically a distribution, the goal of variational methods is to find a so-called variational distribution that is both similar to that of the original distribution and easier to work with. Using Jensen's inequality, the likelihood of this new variational distribution can be considered a lower bound on the likelihood of the original distribution.

Closeness of the variational distribution to the original one, or lack thereof, is most often measured as the Kullback-Leibler (KL) divergence, an information theoretic metric with interesting properties relating to probabilities. It is given by:

$$D_{KL}(Q \parallel P) = \int Q(\theta) \log \frac{Q(\theta)}{P(\theta)} \, d\theta \quad . \tag{2.5}$$

In the case of LDA, a way to make the posterior distribution easier to work with is to relax some relations between the hidden variables (which are pictured as edges in Fig. 2.1). Indeed, the newly defined variational distributions, $Q(\theta)$, $Q(\beta)$ and $Q(z)$, will not depend on each other nor on the corpus:

$$Q(\theta, \beta, z) = \prod_{k=1}^{K} Q(\beta_k) \prod_{m=1}^{M} Q(\theta_m) \prod_{n=1}^{N} Q(z_n) \quad .$$

Instead, the influence of their relations will be reflected in the choice of the parameters for these variational distributions. In the case of LDA, the variational distributions are defined as follows:

- $Q(\theta_m) \sim Dir(\gamma)$, with γ a K-length vector of strictly positive values;
- $Q(\beta_k) \sim Dir(\lambda)$, with λ a V-length vector of strictly positive values;
- $Q(z_{mn}) = \phi_{mn}$, with ϕ_{mn} a K-length vector of values on the K-dimensional simplex. The value of ϕ_{mn} is specific to the n^{th} word of the m^{th} document. Going forward, we will let ϕ_m denote the $N_m \times K$ matrix containing the N ϕ_{mn} vectors of length K specific to the m^{th} document.

Finding the optimal values for these variational parameters γ, λ, and $[\phi_1, \ldots, \phi_M]$ is the focus of the VI algorithm. Because these optimal values will minimize the KL divergence (see equation 2.5) between the variational and the original posterior, it is possible to derive updates to use in an EM algorithm, such as discussed in Section 1.4. We define the following notations:

- Ψ is the digamma function (the first derivative of the logarithm of the gamma function);
- v is such that, for a given word vector w_n, $w_{nv} = 1$;
- γ^*, λ^*, and $[\phi_1^*, \ldots, \phi_M^*]$ are the latest optimal values for the variational parameters.

The steps of the actual computation are then as follows:

1. E-step: The VI algorithm is applied to find optimal values of the variational parameters $[\phi_1, \ldots, \phi_M]$, γ and λ. The steps are, for each document d_m with $m \in [1, \ldots, M]$:
   (a) For each word w_n with $n \in [1, \ldots, N_m]$, being the v^{th} word of the vocabulary:
      i. For k in $[1, \ldots, K]$:
      Update ϕ_{mnk}, the k^{th} element of ϕ_{mn}, using the following formula:
      $$\phi_{mnk} \propto \beta_{kv} \exp \Psi(\gamma_k^*)$$
      ii. Normalize ϕ_{mn} to sum to 1;
   (b) Update γ using the following formula:
   $$\gamma = \alpha + \sum_{n=1}^{N_m} \phi_n^*$$
   (c) Update λ using the following formula:
   $$\lambda = \eta + \sum_{m=1}^{M} \sum_{n=1}^{N_m} \phi_m^* w_{mn}$$
   (d) Repeat until convergence.
2. M-step: Using these variational distributions with their optimal parameters, one maximizes the lower bound on the likelihood of the model, with respect to α and η.

VI is a powerful method, and has been extended to the online variational algorithm for LDA, or simply online LDA (Hoffman et al., 2010). The main contribution of this implementation is the ability to stream documents into the algorithm one at a time. This allows to avoid keeping the entire corpus in memory, which is very important for scalability. Online LDA also allows to add the information of new documents to an already trained model.

2.5 Author-Topic Model

One extension of the LDA framework that is relevant to this work is certainly the Author-Topic Model (ATM) (Rosen-Zvi et al., 2004). In this paradigm, one considers that information is available about the possibly multiple authors of every document. The main difference with vanilla LDA is that, instead of attributing a mixture of topics to the documents directly, the mixture of topics are attributed to the authors of the documents. The generative process is therefore changed so that, assuming equal contribution, for every word one of the authors of the document is drawn and only then is a topic drawn from this specific author's mixture of topics (Rosen-Zvi et al., 2004).

To formalize this, we will use the basis of the LDA process with some tweaks. In addition to the known quantities detailed (see Section 2.2), we now take into account the following:

- L, the number of possible authors.
- A, an $M \times L$ matrix that, for each of the M documents, represented as rows, has value 1 on the columns of its authors and 0 everywhere else. We later refer to the row pertaining to the m^{th} document as A_m.

Using these values, one can once again sample from probability distributions to characterize the corpus:

- Sample M times from $N \sim Pois(\zeta)$, yielding the length of each document, i.e. an M-length vector of integers. We later refer to the length of the m^{th} document as N_m.
- Sample L times from $\theta \sim Dir(\alpha)$, yielding the probability of each topic for each author, or mixing proportions, i.e. a total of L vectors of length K forming an $L \times K$ matrix. We later refer to the mixing proportions of the l^{th} author as θ_l.
- Sample K times from $\beta \sim Dir(\eta)$, yielding the probability of each word for each topic, i.e. a total of K vectors of length V forming a $K \times V$ matrix. We later refer to the word probabilities of the k^{th} topic as β_k.

Here, the main difference with LDA is that instead of attributing mixing proportions for documents, we attribute them to authors and therefore have L different mixing proportions. It follows logically from the intuition that each author must have their own set of favorite subjects. After that, the actual generative process is as follows. For each of the M documents indexed by m, for each of its N words indexed by n:

1. Sample once from $a_n \sim Multinom(1, A_m \|A_m\|^{-1})$, yielding an L-length unit vector having 1 as the l^{th} value, which is equivalent to picking an author at random. This means that the present topic z_n will be chosen with the probabilities specific to the l^{th} possible author.
2. Sample once from $z_n \sim Multinom(1, \theta_l)$, yielding a K-length unit vector having 1 as the k^{th} value. This means that the present word w_n will be chosen with the specific probabilities of the k^{th} topic.
3. Sample once from $w_n \sim Multinom(1, \beta_k)$, yielding a V-length unit vector having 1 as the v^{th} value. This means that the presently chosen word w_n is the v^{th} of the vocabulary.

The computation of the posterior $P(\theta, \beta, a, z \mid \mathcal{C})$, as for LDA, is made complex by the coupling between hidden variables. Fortunately, the solutions we detailed before have easily derived implementations for ATM as well.

The ATM has implications outside the simple inference of topic preferences for the authors of scientific articles. Indeed, the added level in the generative model can readily be used to represent any categorical influence over topic choice. For instance, the "authors" variable can be substituted with the persons who visited a site of interest (Jiang et al., 2015), or with a treatment in a pharmacological trial (Chung et al., 2015).

Chapter 3

Non-negative Matrix Factorization

3.1 Intuition

Non-negative Matrix Factorization (NMF) is another technique that is, in practice, often used for topic modeling purposes. It has the properties that it only gives non-negative components adding up in varying proportions to represent the original values, thus simplifying interpretation in some cases (Lee & Seung, 1999; Y.-X. Wang & Zhang, 2013). Furthermore, it does not rely on parametric distributions, which has upsides in some instances such as better performances in small corpus. Indeed, with fewer observations, probabilistic models can become at risk of having overly influential chosen priors, which can eventually lead to surprising results.

Here C, an $M \times V$ matrix having documents in vectorized forms as rows and words as columns, will be approximated by the product of two smaller matrices W of size $M \times K$ and H of size $K \times V$ (see equation 3.1). The specificity of NMF is that all the values of the factor matrices are constrained to be non-negative, which means that any entry of the original matrix C, indexed for row m and column v, can be approximated by an addition of its components (see equation 3.2).

$$\tilde{C} = W \times H \tag{3.1}$$

$$C_{mv} \approx \sum_{k=1}^{K} W_{mk} \times H_{kv} \tag{3.2}$$

This has many beneficial effects, such as stronger meaning for components in contexts where negative values or probabilistic interpretations do not make sense. In the case of machine vision, NMF has been suggested to obtain a more localized decomposition of pictures which is more coherent with the way our brains work (Y.-X. Wang & Zhang, 2013).

3.2 Loss functions

NMF computation is typically formulated as a numerical optimization problem because, given the positivity constraint, there is often no analytic solution. This implies that one needs to choose both the loss function, also called cost or objective function, and the optimization algorithm.

The two most common objective functions for NMF are the squared Euclidean distance (SED; see equation 3.3) between the original matrix and the approximation, and the generalized Kullback-Leibler divergence (GKLD; see equation 3.4), also called I-divergence.

$$D_{SE}(C \parallel WH) = \sum_{m=1}^{M}\sum_{v=1}^{V}(C_{mv} - (WH)_{mv})^2 \tag{3.3}$$

$$D_{GKL}(C \parallel WH) = \sum_{m=1}^{M}\sum_{v=1}^{V} C_{mv}\log\frac{C_{mv}}{(WH)_{mv}} - C_{mv} + (WH)_{mv} \tag{3.4}$$

It is worth mentioning that, while the SED is a distance in the mathematical sense with the related properties, GKLD is asymmetric and therefore is not. SED is also typically the loss being minimized in traditional ordinary least squares estimation of linear models.

Finally, it has been observed in practice that algorithms using GKLD can be very slow to converge when the initial scales of the W and X are off (Z. Yang et al., 2011). For this reason, Z. Yang et al. (2011) suggest normalizing the original matrix and using the standard KLD as a loss function, which yields the following loss function:

$$D_{KL}(C \parallel WH) = \sum_{m=1}^{M}\sum_{v=1}^{V} C_{mv}\log\frac{C_{mv}}{(WH)_{mv}} + \log\sum_{m=1}^{M}\sum_{v=1}^{V}(WH)_{mv} \quad . \tag{3.5}$$

Z. Yang et al. (2011) found that NMF computation using this loss computed faster and yielded better approximations. The authors also claim that, although it was previously claimed that GKLD-based NMF and pLSA shared similarities (Ding et al., 2008), the KLD-based version they propose is actually closer as it uses normalized data before computation. Therefore, it would actually optimize the same objective function as pLSA, instead of enforcing similarity by normalizing the component matrices a posteriori as is suggested by Ding et al. (2008).

3.3 Optimization algorithms

The optimization problem resides in minimizing the loss function, while keeping the factor matrices non-negative. Because the minimization of those loss functions is not convex

for W and H jointly (Lee & Seung, 2001), all existing algorithms rely on updating one matrix while keeping the other fixed before alternating. This is, of course, done after a suitable initialization of the factor matrices. Techniques for this may rely on random generation, on SVD or on hierarchical clustering. Depending on the choice of optimization algorithm, this can have a great influence on the final result. Assuming the updating process does not generate negative values, an algorithm for NMF computation could therefore be given as follows:

1. Initialize W and H with non-negative values;
2. While convergence is not obtained and/or the maximum number of iterations is not reached:
   (a) Update matrix H;
   (b) Update matrix W.

The first algorithm proposed to compute NMF relies on iterative multiplicative updates (MU; Lee and Seung, 2001). This approach works with both SED and GKLD losses, and closely resembles that of the EM algorithm. It was first suggested for its ease of implementation and its relative speed (Lee & Seung, 2001). The updates are derived from traditional gradient descent updates, and are shown to yield non-increasing losses. The updates at iteration $(t+1)$ are given below for SED (see equation 3.6) and GKLD (see equation 3.7). It is worth mentioning that, even if this was not done originally, we added a small constant $\tau = 10^{-9}$ in the denominators as an effort to prevent division by zero, as was done by Berry et al. (2007).

$$H_{kv}^{(t+1)} \leftarrow H_{kv}^{(t)} \frac{(W^T C)_{kv}}{(W^T W H)_{kv} + \tau} \qquad W_{mk}^{(t+1)} \leftarrow W_{mk}^{(t)} \frac{(C H^T)_{mk}}{(W H H^T)_{mk} + \tau} \tag{3.6}$$

$$H_{kv}^{(t+1)} \leftarrow H_{kv}^{(t)} \frac{\sum_{m=1}^{M} W_{mk} C_{mv} / (WH)_{mv}}{\sum_{m=1}^{M} W_{mk} + \tau} \qquad W_{mk}^{(t+1)} \leftarrow W_{mk}^{(t)} \frac{\sum_{v=1}^{V} H_{kv} C_{mv} / (WH)_{mv}}{\sum_{v=1}^{V} H_{kv} + \tau} \tag{3.7}$$

As one can see, both pairs of updates use a factor that is reduced to 1 in the case where $C = WH$, which implies that this ideal situation would be permanent. The authors additionally show that these iterative updates lead to non-increasing losses, and claim that the algorithm therefore converges to local minima for both matrices (Lee & Seung, 2001). However, this was later disputed by Gonzalez and Zhang (2005), who showed that this algorithm does not necessarily reach convergence to local minima in most cases, and is virtually unable to do so when working with larger matrices. Moreover, the multiplicative nature of the updates means that elements set to 0 will stay that way, which creates a gradual shrinkage of the possible solutions (Y.-X. Wang & Zhang, 2013). To alleviate this problem and others, one would need to make the optimization problem much more

complex, even if it was originally suggested by Lee and Seung (2001) for its clarity and ease of implementation.

Another alternative would be to use a simple gradient descent algorithm with additive updates. The basic method is to take the derivative of the loss function given the matrix and, step by step, to subtract a small part (called the step size, or the learning rate) of it from the matrix until convergence is reached. This insures that, eventually, a local minimum of the loss function can be reached. In the specific case of computing optimal non-negative factor matrices, this method can be implemented using these simple update rules (see equation 3.8). Here, D stand for the loss function of choice, either SED, GKLD or any other. The values ϵ_H and ϵ_W stand for the step size used in the updates of matrix H and W, respectively. The updates are given as:

$$H^{(t+1)} \leftarrow H^{(t)} - \epsilon_H \frac{\partial D(C \parallel WH)}{\partial H} \quad ; \quad W^{(t+1)} \leftarrow W^{(t)} - \epsilon_W \frac{\partial D(C \parallel WH)}{\partial W} \quad . \tag{3.8}$$

Finally, even if, as Lee and Seung (2001) pointed out, it is simpler to implement, this method is slower to converge in most cases. Moreover, the convergence of such methods is very sensitive to the choice of step size. This can be particularly cumbersome with large datasets, where falling a little short of the optimal step size can have sizable consequences. It is thus no surprise that gradient descent algorithms are an area of expertise in their own right (Ruder, 2017).

Chapter 4

Short text messages

Conventional topic modeling techniques can yield remarkable results on large documents, such as Wikipedia pages or scientific articles. But, it turns out, shorter length documents are ubiquitous and need specific attention for their analysis to yield quality results. Whether they be texts on messaging platforms, micro-blogging posts such as tweets, or feedbacks about products, internet users tend to be concise and to the point. Yet they keep talking, and therefore there must be important trends to unearth from their messages. Furthermore, it is worth mentioning that being able to model the topics of short texts means being able to deconstruct larger documents further, and to derive singular meaning from each of their parts.

If we assume the fact that there is a near-infinite number of these short texts, each containing few words, the main issue with short text topic modeling is self-evident. Using traditional techniques, the rows are very sparse, and it becomes difficult to derive topic models based on word co-occurences. Furthermore, it is our observation that most words employed in short messages are highly frequent ones, with few informative values but that conferred by context. We could remark that this is a fundamental issue for natural language processing: in most cases, we use a few words to say a lot of different things.

From this fact, we can deduce a few things about how short texts should be modeled. For instance, it seems logical that the number of topics used in a document should be a function of its length. If one writes a long text, it becomes more likely that different topics will be addressed. On the other hand, in a single tweet, there should rarely be more than a single explored topic. This can be reflected in the generative process by adapting α, the parameter governing the prior distribution for the topic-document probabilities (discussed in Section 2.2). Indeed, a smaller value of the components of α will lead to a high density towards the edges of the Dirichlet simplex and will more adequately model the situation. It translates directly into the documents being more likely to exhibit a single topic. This can be seen as similar to a mixture of unigrams model (Nigam et al., 2000) except that the possibility of multiple topics per document is retained, albeit small.

Finally, if short texts imply a usually smaller vocabulary and a high sparsity, it would seem logical that rejecting words appearing with a small frequency would somewhat alleviate this issue. This can be done during the stopwords removal phase of the preprocessing, as described in Section 1.2.

4.1 Aggregating documents

One of the most researched way to solve the sparsity issue is to aggregate documents into longer so-called pseudo-documents. The aggregation of short texts was first used by Weng et al. (2010) to build their *TwitterRank* algorithm, whose purpose was to measure topic-sensitive influence of Twitter users. This was mentioned by Hong and Davison (2010), in an effort to empirically assess the best strategy for short text topic modeling. They remark that social media data does not lack criteria upon which to build an aggregation scheme, such as users, hashtags, or participation in a conversation. Furthermore, their findings show the clear advantage of aggregating tweets by author over other methods, such as standard LDA (Blei et al., 2003) or the ATM (Rosen-Zvi et al., 2004). Mehrotra et al. (2013) have found similar results when using hashtags to pool tweets.

Unfortunately, using context clues such as authors or hashtags in order to aggregate does not generalize well. As remarked by Quan et al. (2015), it would be best if the aggregation factor came from the short texts themselves. To further this argument, they proposed a self-aggregation based topic model (SATM) scheme in which clustering is used prior to any topic modeling. This is done implicitly by adding a final step to the generative process, wherein a long pseudo-document generated as in LDA (Blei et al., 2003).

Unfortunately, as Zuo et al. (2016) observed, Quan et al. (2015) do not assign a probability distribution for words of the pseudo-documents belonging to short documents, which makes the parameters grow linearly with the size of the corpus. As mentioned before, this level of complexity does not offer sufficient description length reduction. Moreover, it makes SATM computation intractable in a timely manner. To cope with this, Zuo et al. (2016) described the pseudo-document-based topic modeling (PTM) scheme. The generative model, as for SATM, takes LDA as a basis. A new Multinomial variable, κ, is introduced to represent the chances of each of the M short texts, denoted by d_m, to belong to any of the J pseudo-documents, denoted by p_j. This Multinomial distribution takes a vector of length M as parameter, which we will denote ι and assume to follow a Dirichlet distribution. We therefore have the following quantities defined before the generative process:

- ζ, the average number of words per document[1];

[1]This is not mentioned by Zuo et al. (2016), but it was added here for the sake of completeness and congruence with the LDA generative process (Blei et al., 2003)

- V, the number of possible words;
- J, the number of pseudo-documents;
- M, the number of short texts;
- K, the number of topics;
- ι, a J-length vector of strictly positive values;
- α, a K-length vector of strictly positive values;
- η, a V-length vector of strictly positive values.

These values allow us to sample from probability distributions to further define the corpus:

- Sample M times from $N \sim Pois(\zeta)$, yielding the length of each short text, i.e. an M-length vector of integers. We later refer to the length of the m^{th} short text as N_m.
- Sample M times from $\kappa \sim Dir(\iota)$, yielding the probability of each short text to belong to each of the pseudo-documents, i.e. a total of M vectors of length J forming an $M \times J$ matrix. We later refer to the chance of belonging to any pseudo-document for the m^{th} short text as κ_m.
- Sample J times from $\theta \sim Dir(\alpha)$, yielding the probability of each topic for each document, or mixing proportions, i.e. a total of J vectors of length K forming a $J \times K$ matrix. We later refer to the mixing proportions of the j^{th} pseudo-document as θ_j.
- Sample K times from $\beta \sim Dir(\eta)$, yielding the probability of each word for each topic, i.e. a total of K vectors of length V forming a $K \times V$ matrix. We later refer to the word probabilities of the k^{th} topic as β_k.

Having this information we can proceed to the building of the corpus itself. For each of the M short texts indexed by m:

1. Sample once from $p \sim Multinom(1, \kappa_j)$, yielding a J-length unit vector having 1 as the j^{th} value. This means that the present short text d_m will be considered to be a part of the j^{th} pseudo-document.
2. For each of the N_m words in the short text d_m:
   (a) Sample once from $z_{mn} \sim Multinom(1, \theta_j)$, yielding a K-length unit vector having 1 as the k^{th} value. This means that the present word w_n will be chosen with the specific probabilities of the k^{th} topic.

(b) Sample once from $w_{mn} \sim Multinom(1, \beta_k)$, yielding a V-length unit vector having 1 as the v^{th} value. This means that the n^{th} word of the m^{th} short text will be the v^{th} word of the vocabulary.

As for LDA and most topic modeling schemes, exact inference is intractable because of the denominator of the posterior, which involves summing over many values. But instead of VI, the authors turn here to a collapsed Gibbs sampling algorithm for efficient estimation. Gibbs sampling is an MCMC method in which one relies on knowing conditional distributions to generate a random but auto-correlated chain of values, asymptotically constituting a sample from the targeted distribution. This technique is widely used in scientific fields, and is useful to compute LDA with minimal memory requirements (Griffiths & Steyvers, 2004). In the situation of PTM, the algorithm is called collapsed because it is possible to integrate out θ, β and κ and keep only their conjugate prior parameters in the computations, namely α, η and ι.

PTM is a good framework that puts an intuitive idea, that of pasting together texts too short, into probabilistic terms. It has been investigated in multiple works (Hong & Davison, 2010; Quan et al., 2015; Yin & Wang, 2014; Zuo et al., 2016) and has been extended to use different clustering criteria (Bicalho et al., 2017). Unfortunately, there are little to no good implementations of this method in popular data science programming languages.

4.2 Using co-occurences directly

Another suggested method to handle the sparsity issue of short text documents was looking at word co-occurences, or biterms, directly (Yan, Guo, Lan, et al., 2013; Yan, Guo, Liu, et al., 2013). Indeed, topic modeling is essentially the act of finding which words are frequently co-occuring across the documents and explaining it with latent unobserved topics. Therefore, the intuition was this: it is possible to use a square, symmetric, less sparse word co-occurences matrix to learn useful topic models. Of course, in such a scheme it wouldn't be possible to directly derive the topic mixtures of the documents, but they can still be retrieved a posteriori (Yan, Guo, Lan, et al., 2013; Yan, Guo, Liu, et al., 2013).

In this context, the corpus is not to be seen as a collection of documents anymore, but as a collection of relations between two words. These relations are stronger or weaker, depending on the number of documents two words have in common. This, remarkably, makes the corpus an undirected weighted graph, with words as nodes and number of co-occurences as edges, which can be adequately summarized by a contingency matrix. This idea was first mentioned as a way to address short texts sparsity by Yan, Guo, Liu, et al. (2013), in a work where they used symmetric non-negative matrix factorization (SMNF) on a co-occurences matrix.

Since the first times SMNF was first discussed (Catral et al., 2004; Ding et al., 2005), it has mainly been used in graph clustering for community detection (Du et al., 2017; Kuang et al., 2012; X. Shi et al., 2015) with, for instance, some remarkable applications in neurosciences (Li et al., 2018). But from its inception, it was also shown to be useful in summarizing a corpus without retaining information about its documents (Y. Wang, 2008) or to cluster documents according to the words they contain (Kuang & Park, 2013). In the specific case of short text topic modeling, using SNMF on a co-ocurrences matrix yielded meaningful topics (Yan, Guo, Liu, et al., 2013).

This prompted Yan, Guo, Lan, et al. (2013) to propose a new probabilistic model, closer to LDA but founded on the same vision of the corpus as a graph of interconnected words, which they called the biterm topic model (BTM). Here, a biterm is an exchangeable set of two different words co-occuring in a document, which we denote $b = \{w_1, w_2\}$. The corpus, represented as a symmetric contingency matrix containing the weights of the biterms, i.e. the number of co-occurences of their words, will be denoted B. In keeping with other probabilistic topic models, they offered the generative model of a corpus using biterms. The following values are assumed known:

- V, the number of possible words;
- K, the number of topics;
- I, the number of biterms in the corpus;
- α, a K-length vector of strictly positive values;
- η, a V-length vector of strictly positive values.

Using these values, we can then sample from several probability distributions to get more information about the future corpus:

- Sample K times from $\beta \sim Dir(\eta)$, yielding the probability of each word for each topic, i.e. a total of K vectors of length V forming a $K \times V$ matrix. We later refer to the words probabilities of the k^{th} topic as β_k.
- Sample one time from $\theta \sim Dir(\alpha)$, yielding the probability of each topic for the whole corpus, i.e. a vector of length K.

Then, for each of the I biterms indexed by i:

1. Sample once from $z_i \sim Multinom(1, \theta)$, yielding a K-length unit vector having 1 as the k^{th} value. This means that b_i, the i^{th} biterm of the corpus, will be chosen with the specific probabilities of the k^{th} topic.
2. Sample twice from $w \sim Multinom(1, \beta_k)$, yielding the two words w_{i1} and w_{i2} constituting b_i.

As we can see, the individual documents do not have a place in this process, and as such their topic mixtures are not estimated with the probabilities of words per topic but rather deduced after the fact. This is done empirically, using the following intuition:

$$P(z \mid d) = \sum_{i=1}^{I} P(z \mid b_i) P(b_i \mid d) \quad .$$

As usual, estimating this model implies obtaining the posterior distribution of the Dirichlet parameters, β and θ, which is made intractable by a sum over all words and topics in the denominator. In their presentation of the BTM, Yan, Guo, Lan, et al. (2013) suggested the use of a Gibbs sampling algorithm citing as reasons the reduced memory needs and the asymptotic properties, which they argue make it a better alternative than VI, for instance. As explained before, the goal is to estimate the targets by sampling repetitively the known conditional distributions, using past values each time. Here, as for PTM (see Section 4.1), a collapsed Gibbs sampling algorithm is used (Griffiths & Steyvers, 2004). In this specific case it means that, due to the Multinomial and Dirichlet distributions being conjugate, θ and β can be integrated out of the posterior.

The proposed algorithm assumes we know K, α, η and the corpus B, and mandates that we define:

- Z^b_{-i}, a $K \times (I-1)$ matrix containing the topic assignments for each of the I biterms except the i^{th} one;
- Z^w, a $K \times V$ matrix containing, for each value Z^w_{kv}, the number of times the v^{th} word has been sampled using the k^{th} topic;
- Z^b, a $K \times I$ matrix containing, for each value Z^b_{ki}, the number of times the i^{th} biterm has been sampled using the k^{th} topic.

In BTM, α and η are considered to have equal values for all their elements, which leads to symmetric Dirichlet distributions. We therefore denote with $\dot{\alpha}$ and $\dot{\eta}$ the values each of their respective elements take. Knowing this, we can then describe the sampling algorithm with the following pseudo-code:

1. Initialize topics randomly for all biterms;
2. While the maximum number of iterations is not reached;
   For each of the I biterms b indexed by i, made of the words w_1 and w_2 respectively the ${v_1}^{\text{th}}$ and the ${v_2}^{\text{th}}$ words in the vocabulary:

(a) Sample from $z \sim P(z \mid Z^b_{-i}, \alpha, \eta, B)$, which is easily found using basic probability properties:

$$P(z \mid Z^b_{-i}, \alpha, \eta, B) \propto (Z^b_{ki} + \dot{\alpha}) \frac{(Z^w_{kv_1} + \dot{\eta})(Z^w_{kv_2} + \dot{\eta})^2}{\|_1 N^w_k\|_1 / M\dot{\eta}}$$

This yields the K length vector z with 1 as the k^{th} value and 0 elsewhere.

(b) Update Z^b_{ki}, $Z^w_{kv_1}$ and $Z^w_{kv_2}$.

3. Compute β and θ (which in this case is a vector of K values containing the mixing proportions for the whole corpus), using the following formulas:

$$\beta_{kv} = \frac{Z^w_{kv} + \dot{\eta}}{\|Z^w_k\|_1 + M\dot{\eta}}$$
$$\theta_k = \frac{\|Z^b_k\|_1 + \dot{\alpha}}{I + K\dot{\alpha}}$$

Obviously BTM has some caveats and removes a fair amount of complexity from the issue, but it remains an efficient method which has the merit of looking at the problem from another angle, drawing from the fields of both topic modeling and graph theory.

4.3 Using distributed representations

Recent research has also focused on another prospect which shows promise; that of using distributed representations (DR), or word embeddings, to represent documents and avoid the sparsity issue (Bicalho et al., 2017; Gao et al., 2019; Nguyen et al., 2015). The idea is to represent words not with a highly sparse indicator vector (as was done until here), but with a shorter vector of real values that hold semantic information about the word and its relation to others. In short, to have the representation of the word say something about the word, and to have these representations be similar for words that mean the same thing. Using these DR, it is perfectly possible to find that the operation $w_{king} - w_{man} + w_{woman}$ yields a vector that is most close with w_{queen} in real use cases.

Most of the time, DR are obtained by training V single-layer neural networks to recognize each of the V words by its neighbors, using a reliable sample of language as training data. The weights of this single middle layer, of which there are a number R usually such that $R < V$, will become the DR. Indeed, the funding principle behind this technique is the same as for most natural language processing techniques, famously put into words by the linguist Firth (1957): "You shall know a word by the company it keeps". We will not focus anymore on how these DR are obtained, as this subject is vast enough to deserve several theses on its own right. Nevertheless, DR allow to convert a high-dimensional, highly sparse corpus into one that is denser and contains information from an external corpus.

It is possible to train LDA (or a mixture of unigrams model) on these externally trained DR, or latent features as they are called by Nguyen et al. (2015) in their work. Indeed, the authors show excellent LDA results on tweets when documents are vectorized using DR trained on large well-known corpora instead of just using count representation of documents. To be fair, DR of words and their averaging in order to represent documents could also be seen as some kind of topic modeling in itself, except for the fact that DR are not created using the basis of documents co-occurences but using a moving fixed window, taking in a streaming corpus. There is nothing in the training of DR that makes their dimensions distinguishable from one another, whereas the idea at the very core of LDA (Blei et al., 2003), for instance, is that each word is drawn from a single topic, with the probabilities of these topics being on the K-dimensional simplex. Nevertheless, the idea itself is very close, and further raises the question of what constitutes topic modeling.

Finally, it is interesting to notice that all these options to address short text sparsity have some common themes. It appears, for instance, that each of them involves a way to bypass the original structure of the corpus whether it be by grouping documents, discarding them and keeping only the co-occurences of the words, or converting them using externally trained DR. Bicalho et al. (2017) have proposed a way to use some of these methods conjointly. Leveraging the view of documents as points in a high-dimensional space, they suggest clustering short documents into pseudo-document based on their proximity in the space defined by either word co-occurences, or externally trained DR.

Chapter 5

Evaluation of topic models

In order to correctly assess the performances of the showcased algorithms, we will need to define certain metrics. First and foremost, the semantic coherence and interpretability of the topics will be discussed in each use case. Nevertheless, it is also important to have numerical approximations of the fits of the models. Indeed, if human interpretability is a desired aspect of topic modeling, it is not its only purpose. Very often, topic modeling is used to reduce dimensionality in a complex dataset while retaining a maximum of information. Unfortunately, maximizing the retained information does not imply, or even correlates with, maximizing the meaning of the extracted topics (Chang et al., 2009). It is indeed a frequent disappointment in machine learning, to find that the true patterns in data are under no obligation to be meaningful to humans.

We also note that, because of this, it is common in the literature to evaluate a topic model by using its topics as features in a predictive model. It is useful to compare different topic models, assuming a fixed setting, and it can help to show how useful the topics are in a real use case. Furthermore, it allows using classification metrics that may be better known that those we will describe here. We will, however, refrain from using this extrinsic kind of evaluation because it can add confounding factors into the mix, and it would involve too much theory from other fields of data science such as classification models and their evaluation.

5.1 Perplexity

A common and intuitive way to evaluate a model is to look at how well it fits the data or, phrased differently, how much of the observed variance it can account for. Unfortunately, there always exists a risk that a model will fit too tightly, and will not be useful in understanding data outside its training set. When that happens, the model is said to overfit. In machine learning, the usual way to address this issue is to cross-validate results. This is typically done by holding out some of the training data for evaluation purposes, but other cross-validation processes are possible where each part of the dataset is used at least once

for independent evaluation. This allows checking whether the model is able to predict an outcome based on new data or, in the case of topic modeling, adequately summarize a new corpus.

Naturally, a common metric to evaluate LDA and other topic models is simply the probability of a new corpus given the estimated model. This metric is sometimes called *perplexity*. A way of thinking about it is as a measure of how surprised, or *perplex*, the model gets upon seeing new data. Assuming the estimated parameters are those of standard LDA (as detailed in Section 2.3), with $\mathcal{C}$ a held-out corpus containing M documents d_m each containing N_m words, we can formalize perplexity as follows:

$$\begin{aligned} \mathrm{Pe}(\mathcal{C}) &= \exp\left[-\frac{L(\mathcal{C})}{\sum_{m=1}^{M} N_m}\right] \\ L(\mathcal{C}) &= \log P(\mathcal{C} \mid z, \beta, \alpha) \\ &= \log \frac{P(\beta, \alpha, z, \mathcal{C})}{P(\beta, \alpha, z)} \\ &= \log \frac{P(\beta, \alpha, z, \mathcal{C})}{\int P(\beta, \alpha, z, \mathcal{C})\, d\mathcal{C}} \quad . \end{aligned} \tag{5.1}$$

Because the probability of a different corpus is estimated, we do not need to use θ, the document-topic probabilities that were found during training, but just the hyper-parameter α. A separate generative process is assumed for the held-out corpus. However, the exact computation of the likelihood of a document is intractable as it would once again involve normalizing over many possible values for each of the parameters (see equation 5.1). We note that the same issue presented itself for the estimation of posterior probabilities in many other probabilistic models (see Section 2.3, for instance). Perplexity must therefore be estimated, which can be done in different ways. Special attention was given to this issue by Wallach et al. (2009), who reviewed many common techniques and endeavored to show that those were likely to be inefficient and/or inaccurate. To remedy this, they also suggested new methods of their own design to estimate perplexity.

5.2 Word intrusion

Chang et al. (2009) remarked that most topic modeling papers only relied on perplexity or external task utility as means of measuring the success of the methods, overlooking some fundamental aspects such as human interpretability. Furthermore, when interpretability is mentioned, it tends to be described as good or bad without empirical evaluation. In an effort to formalize the assessment of this aspect, they proposed two tasks for human examiners based on the concept of *intrusion*.

In the task of word intrusion, the subject will be shown six randomly ordered words. Among these words, five were taken from a topic z_1 with which they were highly associated, and the last one was chosen because it had a low association with topic z_1, but a high association with another topic z_2. If the topic z_1 is semantically coherent, the subject should be able to pick the intruder word with a high probability. Therefore, word intrusion retrieval measures the strength of the semantic association between words and topics. Similarly, they proposed a task called topic intrusion, during which the subject is shown a document and four topics, described by their highest ranking words. Three of those topics are the highest probability topics assigned to the document, while the last is an intruder chosen for having a low association with the document. It follows that, if there is good semantic association between a document and its high ranking topics, the subject should be able to pick the intruder with a high probability.

However thorough this approach may be, it still demands human implication which is costly and unscalable. For this reason, Lau et al. (2014) automatically assess the word intrusion detection probability. They used for this task the corollary of the methodology they developed to assess the most representative word in a topic with a ranking Support Vector Machine, using different indicators such as pointwise mutual information (PMI) or original topic-word probability as features (Lau et al., 2010).

5.3 Coherence

As mentioned in Section 5.2, likelihood is often used as a proxy for topic interpretability when it is, in fact, not even positively correlated with it (Chang et al., 2009). However, as made evident by the intrusion experiments of Chang et al. (2009), whether a topic makes sense to humans seems to be related to how similar its "top words" are. Indeed, in a strong topic model, one would spontaneously expect to see some form of vocabulary clustering take place. Each topic would then have the highest probability to spawn words that are closely related independently of context, that are *coherent* with each other.

5.3.1 Extrinsic evaluation

A first attempt at measuring this form of topic coherence was made by Newman et al. (2010). The basic concept is to examine the set of top words for each topic, $\mathcal{W}_z$, i.e. the words for which $P(w \mid z)$ is highest, and investigate how likely they are to be associated independently of the present context.

This can be done by taking the mean (or median) of some symmetric similarity measure $S(w_1, w_2)$ for all exchangeable combinations among the top words. The number of top words being considered is a hyper-parameter. The general formula for coherence is then as follows:

$$\mathrm{Co}(z) = \mathbb{E}_{w_1,w_2 \in \mathcal{W}_z} S(w_1, w_2)\mathbf{1}_{\{w_1 \neq w_2\}} \quad .$$

When used with PMI as similarity measure (see equation 5.2), the formula gives what has become known as the UCI coherence metric, named after David Newman's university. Indeed, Newman et al. (2010) showed that coherence computed with PMI was highly correlated with human judgment. It is also worth mentioning that, in order to obtain a robust metric that is on a fixed interpretable scale, PMI can be normalized as described by Bouma (2009) to yield the NPMI (see equation 5.3).

$$\mathrm{PMI}(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)} \tag{5.2}$$

$$\mathrm{NPMI}(w_1, w_2) = \frac{\mathrm{PMI}(w_1, w_2)}{-\log P(w_1, w_2)} \tag{5.3}$$

In practice, one uses an external corpus to derive the marginal probabilities of words, $P(w_1, w_2)$, $P(w_1)$ and $P(w_2)$, in a context-independent way. In the work of Newman et al. (2010) different external corpora are considered to compute these frequencies, but using the articles of Wikipedia seemed to yield the best correlation between coherence and human judgment.

Another interesting option first presented by Newman, Karimi, et al. (2009) was to use the number of hits (or its logarithm) for a web search engine query containing the top words as a similarity metric, instead of PMI. Indeed, a Google search for "machine learning" yields 2.09 billions results, whereas a search for "machine rabbit" only yields 95.1 millions. This straightforward method gives a strong indication of how common word groupings are and is independent of domain, language, or even time. It is, however, made difficult to use in practice by the limitations search engines place upon the free use of their API.

5.3.2 Intrinsic evaluation

Another way to evaluate topic coherence without resorting to external corpora was proposed by Mimno et al. (2011). It has become known as the UMass coherence measure, after David Mimno's university. The rationale, based on expert annotations of topics, is that most topic issues could be made evident by checking whether frequent words are good predictors of less frequent words inside the corpus. To formulate it properly, we define a function $F(w)$, taking a word as input and giving its document frequency, i.e. the number of documents in which the word appears at least once. Similarly, $F(w_1, w_2)$ is simply the number of documents in which w_1 and w_2 both appear at least once. We then have the following coherence formula:

$$\mathrm{Co}_{\mathrm{UMass}}(z) = \sum_{w_r, w_f \in \mathcal{W}_z} \log \frac{F(w_r, w_f) + 1}{F(w_f)} \mathbf{1}_{\{F(w_f) > F(w_r)\}} \quad .$$

It is easily noticed how similar the formulation is to that of the UCI coherence. Notwithstanding the lack of dependence on an external corpus, the main difference lies in the similarity measure. As one can see, the logarithm of a (smoothed) conditional probability is used, which is not symmetric. The probability is always that of the rarer word conditioned on the more frequent word. Indeed, Mimno et al. (2011) have found this metric to be slightly more correlated with human judgment than PMI. Finally, these similarity measures are summed instead of averaged. It is a powerful tool, easy to implement with no reliance on external data.

Part II

Applications

Chapter 6

Experimental setting

Throughout the rest of this thesis, we detail multiple analyses we conducted in a consistent environment, while making different design choices. For reproducibility, it is essential to now present some aspects of this environment and these choices.

6.1 Models

First and foremost, we have decided to examine only four different topic models, to exemplify different existing strategies.

LDA (Blei et al., 2003) As stated before this is a very well known model that has spawn many extensions. We include it to compare its results, which could be considered standard in the field, with other techniques that are more fine-tuned to work well with short documents.

NMF (Lee & Seung, 1999) A non-probabilistic model, NMF is included for contrast with the other three models that do fall in this category. Furthermore, its famed ability to generate sparse components may yield good results with short documents.

BTM (Yan, Guo, Lan, et al., 2013) Specifically created to alleviate the sparsity issue in short documents, the BTM offers a probabilistic way to analyze the corpus as a network of words rather than a collection of documents.

PTM (Zuo et al., 2016) Exemplifying another solution to the sparsity issue, the PTM is a complete probabilistic model that does not rely on external aggregating features and that correctly addresses the shortcomings of the earlier SATM (Quan et al., 2015).

There are multiple models we chose to include. The ATM (Rosen-Zvi et al., 2004) was not selected, as it was shown by Hong and Davison (2010) to not adequately modeling tweets. Using SNMF on word correlations (Yan, Guo, Liu, et al., 2013) also seemed like a

very promising idea, but we unfortunately did not find an useable implementation of this algorithm.

6.2 Parameters

As there are numerous parameters the effect of which one could study, one must make some choices and present them. The number of topics K, in topic modeling, is typically assumed fixed and left for the user to specify or optimize, even though there exist ways to automatically determine the best one (Zhao et al., 2015) or to circumvent the issue entirely (Teh et al., 2006). Instead of using these solutions or grid-searching for the best K, we will focus on showing how the scores and the interpretability of models evolve with the amount of topics.

Because our first aim is to investigate different types of models, we must be especially restrictive regarding hyperparameter tuning. Varying K for our four chosen models will already yield multiple interesting comparisons, which we deem numerous enough to be informative without being too complex. For this reason, we refrain from varying other parameters and instead choose to make consistent assumptions. Firstly, we assume the prior values of both α and η (see Section 2.2) to be equal to $1/K$, which is an arbitrary but reasonable default. There exist better ways to determine these values, but these fall beyond the scope of our work and have no real translation in the framework of NMF. For PTM, a number of pseudo-documents J must be specified prior to modeling topics. As Zuo et al. (2016), in their experiments, set K to 100 and J to 1000, we chose to set $J = 10K$ for all PTM computations.

6.3 Programming

Since hardware choices influence how software runs, it seems important to mention that all computations are done on a personal Lenovo ThinkPad T14, a machine running a 64-bit Arch Linux distribution with the 5.13 Linux kernel. The computer has 16 GB of memory and an AMD CPU with 16 logical cores.

We will be using Python (Van Rossum & Drake Jr., 1995) for all purposes, though it may be worth mentioning that some of the models we use were coded in other programming languages but are provided with Python interfaces in any case. We chose Python since it is a widely known open-source, high-level programming language that is both powerful and flexible. Furthermore, Python has bindings with many APIs, machine learning frameworks and visualization toolboxes, which contributes to make it a strong choice for any data science project. In our specific case, we use Anaconda 3, a distribution of Python coming with the version 3.8.11. Well-known in the data science field, Anaconda provides a flexible

but robust way to manage dependencies between packages while providing a stable, up-to-date ecosystem at all points. For the actual model computations, we rely on different open-source Python packages, making the reproducibility of our results easier.

Tomotopy Tomotopy (bab2min et al., 2021) is a relatively new Python package, as its development started in May 2019. It aims to provide implementations for numerous topic models, with deliberate algorithm choices and a C++ backend for speed. It is the only package we could find that implemented the PTM and, for the other available models we tested, it gave very fast results. Tomotopy also has a lot of useful tools to extract results or work with topic models in general. The only downside of this package is a very common one; for the sake of simplicity, the authors assume that all pre-processing will be done by Tomotopy. This is easier when the goal is to train different topic models quickly, but it creates issues when one would like to compare results with other models that are not available in the package. In our case, we chose to use Tomotopy not only for PTM but for LDA as well, a model for which numerous other implementations were available. In this framework, LDA is computed using the distributed Gibbs sampling algorithm developed by Newman, Asuncion, et al. (2009). Regarding PTM, the authors only state that they follow the original description (Zuo et al., 2016) where a collapsed Gibbs sampling algorithm was proposed.

Scikit-Learn Scikit-Learn (Pedregosa et al., 2011) is a general purpose statistics and machine learning toolkit, including valuable tools for preprocessing, cross-validation and evaluation. It has an extensive documentation, and the interface of its models is widely used as a default when proposing new machine learning applications. Indeed, it is often seen that new model implementations will mimic the code style and interfaces of Scikit-Learn models as a way to standardize. Scikit-Learn offers some implementations of topic models, of which LDA and NMF are of interest to this work. Unfortunately, there are some unresolved issues with their implementation of LDA, which made us wary of using it here. However, we decided to use their NMF model as it is fast and reliable. Their implementation allows using either a multiplicative update or a gradient descent algorithm for computation, as well as choosing the loss function between SED and GKLD. We decided to use the gradient descent algorithm, as it was more stable, in combination with the SED loss function which was the default.

Bitermplus Bitermplus (Terpilowski, 2021) is a very recently developed Python package implementing BTM. More specifically, it aims to provide Python bindings for the original C code of Yan, Guo, Lan, et al. (2013). As such, it uses the Gibbs algorithm described in the original article. It is still in active development, but it already comes with different tools to evaluate models and analyze results. Its implementation of perplexity computation, for instance, has heavily influenced ours. Bitermplus is our best choice to include a sound BTM implementation in the comparison. However, it should also be noted that this implementation is not very robust in the face of large datasets or unusual values, that it

is by far the slowest of our chosen models and that, as Tomotopy, it forces the use of an unusual preprocessing pipeline. Fortunately, all those downsides were manageable.

Finally, we believe that reproducibility and transparency are of crucial importance to science, technology and society as a whole. To uphold those values, we make all our code publicly available along with this thesis. It is free of use, modification and distribution. In order to guarantee the reproducibility of the results, we have also fixed random seeds wherever stochastic processes were involved. Throughout all computations, these seeds ensure that a computer running our code will always generate the exact same results. However, we have made use of parallel computation wherever it worked reliably. This adds uncontrolled randomness in our training and evaluation processes, unfortunately, as the order of the operations can vary between different runs.

6.4 Evaluation

Many topic modeling packages provide implementations of popular evaluation metrics. Unfortunately, those are often either not working as expected or deeply linked with a particular modeling framework. In order to reliably compare the different models, we independently implemented three metrics discussed in Chapter 5.

What these metrics have in common is that they only require an $M \times V$ matrix C, the corpus in a vectorized form, an $M \times K$ matrix θ containing the probabilities of topics given documents, and a $K \times V$ matrix β containing the probabilities of words given topics. In the case of NMF, which is not probabilistic, these matrices θ and β correspond to W and H, respectively, and are normalized to have their rows sum up to 1. It is also worth reminding the reader that, in the BTM, no probabilities are assigned to the mixture of topics in documents, and those are found heuristically after the computation.

Pseudo-perplexity This metric computes numerically the likelihood of the corpus given the model. We use it because, even though it does not inform on how interpretable a model is, it can still give information on the sense it makes on a probabilistic level. This is purely empirical and is computed on the same corpus used for training instead of held-out documents, which is not an issue as we are more concerned with fitting the corpus we have than with generalizing power. Like the true perplexity described in Section 5.1, this metric has a negative relationship with the fit of a model. As the data becomes less likely with regard to the model, the perplexity will become larger and larger.

Our implementation is heavily influenced by that of the Bitermplus package, but did not always work during our tests. The formula of the metric we wrote is simply:

$$PP(\theta, \beta \mid C) = \exp \frac{-\sum_{m=1}^{M} \sum_{v=1}^{V} (C_{mv} \log \sum_{k=1}^{K} \theta_{mk} \beta_{kv})}{\sum_{m=1}^{M} \sum_{v=1}^{V} C_{mv}}$$

As one might notice, one issue is that having $\sum_{k=1}^{K} \theta_{mk} \beta_{kv} = 0$ only once will produce an infinite value, which will make the final result infinite as well. This issue occurred only for NMF, which is not probabilistic and generates sparse component matrices as a central feature. This prompted us, in the case of NMF, to set the null values in its component matrices equal to an arbitrary very small constant (10^{-32}) before normalizing the rows.

Finally, to circumvent not having an assigned topic z for each word, we sum the probabilities of words appearing in a document across all topics. For this reason, we cannot compute perplexity independently for each topic, but only for the model in general. Moving forward we will refer to this metric as "PP".

UMass coherence (internal) We use the straightforward formulation of Mimno et al. (2011), including the use of log-conditional probability as a similarity measure. This metric requires only the corpus C as well as the top words of each topic, easily computed from β. As shown in Section 5.3, this metric is a sum of log-probabilities, which means that it is non-negative and positively related with the goodness of fit of a model.

We set the number of top words to use at 20. As an added value, this metric can be computed independently for each topic and allows us to compare them inside a model. To characterize the whole model, we will simply take the average coherence of all its topics. Moving forward we will refer to this metric as "UC".

Google coherence (external) Using the insight of Newman, Karimi, et al. (2009), we have implemented a last coherence measure without any reliance on the corpus used for training, or even any external corpus except the internet. This algorithm simply takes the top words of a topic, makes a Google search containing all of them (in order) and reports the base-10 logarithm of the number of results. As discussed in Section 5.3, this metric allows us to perform an external evaluation of the topics we found without having to choose an adequate external corpus. Moreover, the resulting number is very straightforward to interpret. It is a non-negative number, usually ranging between 0 and 10. As a high number of Google results would indicate that the top words of this topic are often co-occuring, it is positively related with the goodness of fit of a model.

This is a somewhat more experimental metric, and while it gives us valuable information, its results should still be taken with a grain of salt. For instance, Google does not index the vast majority of tweets, and it indexes in priority those from accounts with many followers. This means that this metric should be considered biased against topics having hashtags

and mentions in their top words. As for UC, we set the number of top words to use at 20. This metric can also be computed independently for each topic. To characterize the whole model, we will take the average coherence of all its topics. Moving forward we will refer to this metric as "GC".

These three metrics, PP, UC and GC, have been chosen to fulfill different roles and offer the best possible comparison of the models we selected. We also selected them because of their prominent use in the literature, and the fact that they allowed us to compare very different models without measuring their performance as features in another model. Indeed, we chose to not use this kind of external validation in order to avoid the additional biases and complexity, as detailed in the beginning of Chapter 5. Moreover, these metrics were possible to implement from scratch, which inspired us more confidence in their ability to compare models. Finally, we have no doubt that a very strong choice to compare the coherence of our different topic models would have been to use human annotators in a word intrusion task (see Section 5.2), for instance. Unfortunately this was not possible, but the coherence metrics we chose have been found to be highly related with human judgment (Mimno et al., 2011; Newman, Karimi, et al., 2009).

Chapter 7

Simulation study

Before using our selected models on real-world data, we designed a simulation study allowing us to compare their implementations in several settings. In this chapter, we will first detail our simulation algorithm, which uses the LDA generative model, and the parameters that we chose to vary. We will then present the different competitors and compare their results across the different settings.

7.1 Generative process

Using the generative model of LDA described by Blei et al. (2003) as a basis, we designed an algorithm capable of creating large simulated corpora with high efficiency. This algorithm was especially created in a way that makes parallel computation possible, which greatly improved its performance. The process of creating a corpus, in this framework, requires the following parameters to be specified:

- M, the number of documents in the corpus, which we arbitrarily fixed at 1000;
- V, the number of possible words in the corpus, which we arbitrarily fixed at 10000;
- K, the true number of topics in the corpus;
- ζ, the average number of words in a document;
- ϵ, the amount of random words to add as noise, as a fraction of the original length of the documents such that if a document as N words without noise it will eventually be of length $N(1+\epsilon)$;
- α, a K-length vector of strictly positive values that we arbitrarily set at $1/K$ for each component;
- η, a V-length vector of strictly positive values that we arbitrarily set at $1/K$ for each component.

Of these parameters, we investigate three. We investigate two values for K, 16 and 64, which we arbitrarily chose as plausible values for low and high numbers of topics. We vary ϵ between 0 and 10%, in order to have settings with and without added noise. Finally, we vary ζ, the most relevant parameter for the purpose of studying topic modeling with short documents, to be either 20 or 1000. We chose these values as adequate average numbers of words for typical short documents, like tweets, and longer document, like news articles. With these three parameters having two possible values each, we have 8 different settings. For each of these 8 settings, we will compute 50 different corpora, which means our final sample size will be of 400 corpora. Having these settings, the generative process follows closely what is described in Section 2.2. The only notables changes are efficiency optimizations and the addition of noise, if any. For instance, for the m^{th} document in the corpus, this would be done by drawing words from $Multinom(\epsilon N_m, 1/V)$ and adding them to the document, which is equivalent to picking words at random without regard for the topics of the document.

After generating each of these 400 corpora, we will fit each of our four chosen topic models. In this case, we know the real value of K, and so we will denote $\tilde{K}$ the number of topics that is used to compute a model. Here, the value of $\tilde{K}$ will be considered a parameter of the models and a function of the real K. We chose to try three values of $\tilde{K}$ for each model: $K/2$, K, and $2K$. Further on, we denote ν the multiplicative factor used to find $\tilde{K}$, such that $\tilde{K} = \nu K$. Finally, we will compare these 4 techniques using 4 values for $\tilde{K}$ across all 8 simulation settings on two metrics defined above: PP and UC. Indeed, because we do not generate real human language but a probabilistic approximation of it, we cannot use GC for this study.

To sum up, we are left after all iterations with $50 \times 8 \times 12 = 4800$ simulated observations, for PP and UC each. Each of these 50 iterations, including the fitting of the models and their evaluation, took on average 58.96 minutes with very little variation. In order to get to this acceptable computing time, we unfortunately had to limit the number of optimization iterations of all models to 25.

7.2 General comparison

We will now examine the results of these simulations and subsequent model computations. For a comprehensive overview, we include large tables with average scores for PP (see Table 7.1) and UC (see Table 7.2). For better visualization of the main results, i.e. the effect of ζ on the performances of each model is also shown in Fig. 7.1. Due to the high number of samples, we chose to use violin plots in order to accurately represent the distributions of each sample. Each violin plot shown in Fig. 7.1 has a sample size of 600 measures, and their horizontal bars represent their extrema and mean.

	ζ	20				1000				
	K	16		64		16		64		
	ϵ	0%	10%	0%	10%	0%	10%	0%	10%	All
Model	ν									
BTM	1/2	1934	2137	578	644	5582	6818	4968	7257	3740
	1	1165	1283	308	342	4647	6184	4587	7021	3192
	2	**640**	**701**	**164**	**180**	4341	5821	4579	6959	2923
LDA	1/2	2879	3203	1223	1417	4334	5399	3186	5061	3338
	1	2139	2408	801	936	3264	4482	2551	4377	2620
	2	1518	1715	512	601	**3014**	**4141**	2072	3790	2170
NMF	1/2	2058	2276	783	880	4988	5970	3290	4556	3100
	1	1287	1460	483	556	3169	4165	1743	2642	1938
	2	933	1060	312	353	3289	4368	**1589**	**2410**	**1789**
PTM	1/2	4292	4651	2535	2824	6701	7676	6082	7620	5298
	1	3545	3862	1414	1600	5808	7076	4815	6786	4363
	2	2631	2918	741	840	4687	6165	5201	6822	3751

Table 7.1: Mean PP scores for the 12 competitors (rows), for 8 different simulation settings (columns), across 50 iterations, with minimum values in bold. Lower values indicate better results.

	ζ	20				1000				
	K	16		64		16		64		
	ϵ	0%	10%	0%	10%	0%	10%	0%	10%	All
Model	ν									
BTM	1/2	869	866	941	941	306	296	431	411	633
	1	905	903	974	972	328	313	445	421	658
	2	**940**	**939**	**1004**	**1002**	337	324	453	426	678
LDA	1/2	851	849	917	913	322	312	445	424	629
	1	880	878	948	943	340	329	459	439	652
	2	909	906	982	977	351	341	472	453	674
NMF	1/2	868	866	937	934	329	322	470	450	647
	1	897	896	962	958	349	341	**485**	464	669
	2	930	928	984	977	**357**	**348**	477	**464**	**683**
PTM	1/2	850	847	917	913	299	294	424	410	619
	1	878	876	947	944	318	311	445	430	644
	2	908	905	981	976	339	330	463	446	669

Table 7.2: Mean UC scores for the 12 competitors (rows), for 8 different simulation settings (columns), across 50 iterations, with maximum values in bold. Higher values indicate better results.

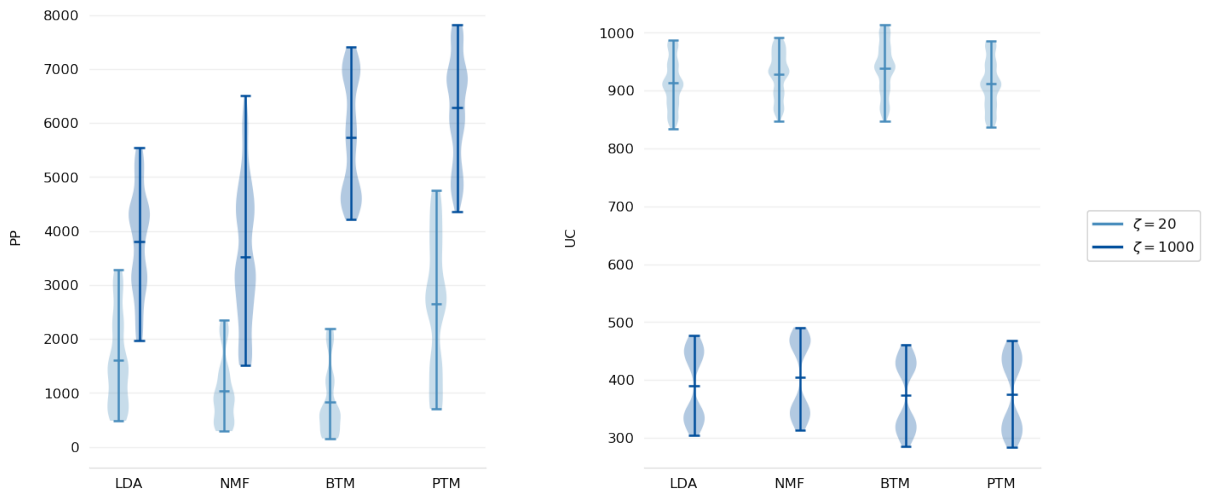

Figure 7.1: Comparison of four models, i.e. LDA, NMF, BTM and PTM (horizontal axis), for $\zeta \in \{20, 1000\}$ (colors), on PP (left) and UC (right)

A first observation we can make is that there is a very clear interaction effect of the model type and ζ on both PP and UC. Indeed, the best models for $\zeta = 20$ do not appear to be the same as those for $\zeta = 1000$. According to both metrics, BTM is the superior model for short documents in all simulation settings, although the contest is close on UC where NMF has good scores as well.

Indeed, according to both metrics, NMF seems to be the best model on longer documents and overall. These results are surprising, as the data was simulated using the LDA generative process. It is very interesting to see that a model with no proper probabilistic interpretation could outperform LDA on finding informative topics, at least according to the metrics we use. Nevertheless, LDA is a close second to NMF for long documents on both metrics.

PTM, however, does not seem to perform very well at all, as it is consistently the worse model regardless of simulation setting or metric. This may be due to the fact that, due to an effort towards limiting the number of factors in this experiment, we did not tune the number of pseudo-documents it uses, and instead let it be equal to $10K$ in all instances. If this number is not optimal, it could unjustly impact the performance of this model.

Unrelated to this we observe that, contrary to what one might expect, the largest value of K is almost always the one yielding the best results. This is not really surprising as none of the metrics aim to evaluate the adequateness of $\tilde{K}$, the true K being unknown in most instances. It also makes sense because increasing $\tilde{K}$ maximizes the amount of information that is retained from the corpus.

The amount of noise we add in the documents, on the other hand, seems to impact negatively the performances on both metrics, although independently of the model. The performances on PP vary the most with this parameter, which could explain the larger dispersion we observe in Fig. 7.1 for PP scores.

Finally, we notice that the effect of the true K is different on PP and UC. Both metrics increase with it, which means that a high count of topics has a negative impact on PP and, interestingly, a positive one on UC. It makes sense that, when a corpus covers more topics, the task of finding them becomes harder, and indeed this is reflected in the relation between PP and true K. We do not, however, have any good hypothesis on why its relationship with UC is positive.

7.3 Relation between metrics

Given our large number of observation, it is possible to describe the relation between PP and UC. A representation of these data is shown in Fig. 7.2, where we discriminate between the two ζ values since the results are of very different magnitudes on both metrics.

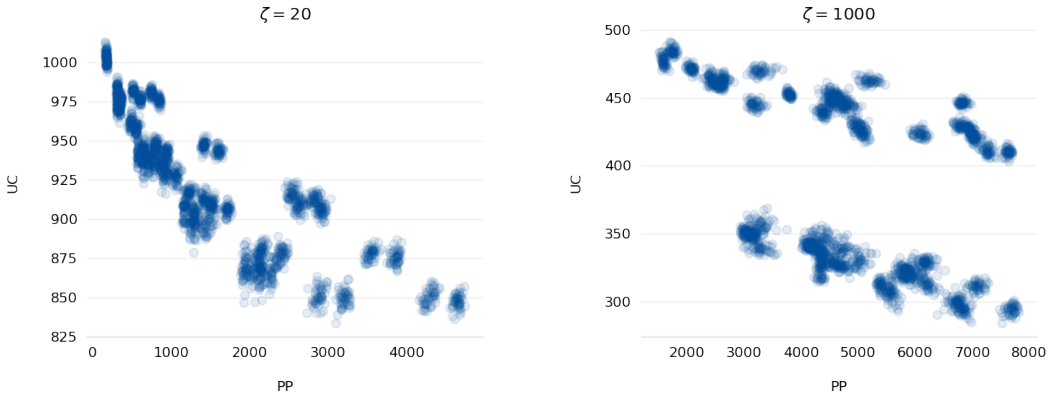

Figure 7.2: Relationship between PP and UC, independently of model or simulation settings, for $\zeta \in 20, 1000$

There seems to be a strong, negative relation between the two, and their Spearman correlation coefficient is overall of $-.897$ ($p < .001$). This means that, overall, a lower PP comes with a higher UC and that in this case they measure similar aspects. This makes sens, although it goes against the findings of Chang et al. (2009). We note that this is true regardless of ζ, but that the relationship seems weaker in the case where $\zeta = 1000$. In that setting, the Spearman coefficient is only equal to $-.525$ ($p < .001$). This seems to be caused by the stronger divide in groups, easily observed in Fig. 7.2 (right). We can attribute this to the relation between UC and the true K, which we discussed in the previous section.

Chapter 8

Application on Twitter data

In this work we set out to examine topic modeling techniques relevant to short text messaging, using simulated data but also with the specific use case of analyzing tweets. In this chapter, we will first present the real world data, then detail the technical choices that were made. We will finally present our results using different techniques that have been proposed to address short text topic modeling challenges, relying on both LDA and NMF frameworks.

8.1 Data

As previously stated, the focus of this thesis is the modeling of short text messages. Whether it be on a micro-blogging platform such as Twitter, in the comments of other social media or in their private messages, it seems that the language of the internet users tends to be short and to the point. With this particular form of speech arise particular challenges.

In order to showcase the tuning of topic modeling techniques specifically to this kind of data, we will use tweets. Publicly available, short and thematic, this appears to be a very good use case for topic modeling. More specifically, we will retrieve and process tweets from June and July 2020 having the hashtag #BlackLivesMatter. It is the author's opinion that the subjects of police brutality and structural racism are of the utmost importance to our moving forward as a society. Black Lives Matter (BLM) is an activist movement against police violence and structural racism, born in 2013 in the US after the police officer who shot and killed Trayvon Martin, an african-american teenager, was acquitted. In May 2020 the murder of George Floyd, an african-american man, by a police officer lead to months of worldwide protests. This grievous event also sparked an intense public debate, where Twitter users exhibited a large array of opinions and concerns. For instance, it is of note that these events unraveled amidst a global pandemic that has serious socio-economic implications, as well as a somewhat disturbed road to general elections in the US.

More objectively, this specific conjunction of protests, demands, accusations, policy proposals and worries, with varying agendas, nuances and levels of discourse, seems to be a very fertile, text-rich ground on which to make topic models grow. Finally, extracting topics from tweets has been a major interest in the literature (Bicalho et al., 2017; Hong & Davison, 2010; Pedrosa et al., 2016; Ramage et al., 2010; Yan, Guo, Liu, et al., 2013; Yin & Wang, 2014), which contributes to make these data an exciting benchmark upon which comparing different techniques.

8.1.1 Scraping

It is common to hear that, with the generalized use of web services, we generate more data than ever, but fetching these data is not necessarily a trivial task. One could almost make the point that large tech companies make every effort to make user consume a constant stream of information, but seem to make it really hard to download, organize and analyze these publicly accessible data.

In this section, we will focus on the scraping of the aforementioned data, i.e. the act of fetching documents from the web and extracting their actionable contents. As there are tools available for extracting the needed documents from Twitter, we will detail how they work and how to use them.

Even though most tweets are publicly available on the website, their automatic retrieval is made complicated by the Twitter API policy of not allowing fetching beyond two weeks ago. Fortunately, a Python package was proposed to remedy this. GetOldTweets3 (Mottl, 2021) cleverly mimics the behavior of a user scrolling through a specific Twitter page and automatically retrieves the JSON files generated by this behavior. Moreover, the package allows to specify different criteria for extracting the tweets, thus making it really simple to only download messages for a specific window in time, for a specific user or containing a specific keyword (such as a hashtag). Using this process, one retrieves a list of observations of the following attributes:

- `id`: String, identifies the tweet uniquely;
- `author_id`: String, identifies the author uniquely;
- `username`: String, the Twitter pseudonym of the author;
- `permalink`: String, a unique link to the tweet;
- `to`: String, the username of the account the tweet was addressed to if it was addressed;
- `text`: String, the content of the tweet;
- `date`: Datetime, an object representation of the timestamp;

- `formatted_date`: String, the date and time of the tweet;
- `retweets`: Integer, the number of times the tweet was shared;
- `favorites`: Integer, gives the number of times the tweet was liked;
- `replies`: Integer, the number of replies to the tweet;
- `mentions`: String, the usernames that were mentioned in the tweet with the @ prefix;
- `hashtags`: String, the hashtags that were used in the tweet;
- `geo`: String, the location from which the tweet was written if available;
- `urls`: String, the web links mentioned in the tweet;

As one can see, it seems easy to build comprehensive data structures from this information. Likes, retweets and replies can be used to quantify the popularity and the controversial aspect of tweets. Hashtags are, in themselves, a way of labeling the topics of a tweet.

Unfortunately, probably as a way of encouraging developers and researchers to use their paid API access, Twitter rejects requests when they exceed a certain amount in an ever shortening window of time. This is the case whether the requests were made via their own free API or via other processes, such as regular HTML requests. To address this, we extracted a maximum of 8000 tweets for each day of June and July, with a 10 minutes break between extractions. The total number of extracted tweets was of 442,433, from 236,538 different authors. Before any processing, there was a total of 247,211 different words in all the tweets.

8.1.2 Preprocessing

The impact of the way text data are processed and transformed before numerical analysis can hardly be overstated. The decisions made in these steps are crucial, and can indeed make or break an algorithm. Nevertheless, the amount of papers overlooking to mention this information when estimating topic models is staggering, and even in articles addressing specifically highly sparse, highly irregular data, these steps may not be discussed (Jipeng et al., 2019; T. Shi et al., 2018; S. Yang et al., 2015; Zuo et al., 2016). In this specific case, both the standard preprocessing that must be performed and the specific challenges of modern social platform messages will be addressed. Finally, the vectorization of the string documents will be discussed before moving on to topic modeling per se.

The necessary preprocessing steps will be taken as follows, for each document in each corpus:

1. Discard all duplicate tweets but one in order to avoid bias coming from bot accounts and spammers.

2. Lowercase all letters and replace some accented or modified forms of letters by their ASCII equivalent to normalize spelling.

3. Find all web links with regular expressions and replace them with a single placeholder.

4. Replace all punctuation marks with spaces, except for # and @.

5. Replace all emojis present in the text with informative placeholders, using the Python package "Emoji" (Taehoon & Wurster, 2021).

6. Tokenize on an unigram basis.

7. Discard all words of length lower than 3 and higher than 36.

8. Discard all words in a list of 127 common, non-informative English stopwords.

9. Consider as stopwords all words that appear in less than 50 documents, and discard them.

10. Finally, remove all tweets that are now empty as a consequence of these transformations.

Removing words based on their frequency or their length is indeed a common step in topic modeling preprocessing (Bicalho et al., 2017; Gao et al., 2019; Pedrosa et al., 2016; Yan, Guo, Liu, et al., 2013; Yin & Wang, 2014). For instance, Yan, Guo, Liu, et al. (2013) used Twitter data and removed all words appearing less than four times in the whole corpus. They subsequently showed that doing so meant the vocabulary size stayed almost the same when increasing the corpus size from 10^4 documents to 10^5.

In our case, all stopwording efforts reduce the size of the vocabulary down to 17,204. Adding stemming to this thorough stopwording, on the other hand, only makes the vocabulary size go down to 12,439. Because it makes interpretation and evaluation (i.e. for topic coherence, where one relies on the words occurrences in other corpora) more difficult, especially in a setting where inaccurate spellings might throw off the stemmer, we chose not to use it. Finally, as a consequence of the whole preprocessing step, the corpus size is down to 385,713. This is due to some tweets being duplicates or containing only stopwords.

As a way of translating the documents into vector representations, we used the `CountVectorizer` class of Scikit-Learn (Pedregosa et al., 2011). Indeed, we chose not to use weighting schemes such as TF-IDF because our documents have highly similar lengths to begin with. In addition, even if weighting words by their occurrences across the whole corpus can sometimes be useful, in this work we prefer to let models react to very frequent words in their own way.

8.1.3 Description

The dataset will now be presented further, which is useful for the later interpretation of the topic models. We present first some characteristics of the tweets, with useful descriptions of their distributions. As one can see in Table 8.1, the vast majority of tweets seem to gain little traction with regard to retweets, replies and favorites. We also observe that the average number of extracted tweets per day is well under 8000 and has some important variations. This is due to the algorithm we used stopping after an arbitrary amount of extractions, sometimes under a thousand, due to Twitter limitations, but fortunately has no bearing on the rest of this work.

We also computed a table containing the 40 most frequent words in the corpus, as well as their probability of appearing in a tweet (see Table 8.2). It appears that the police is a highly common subject, with the hashtag #DefundThePolice being present in 1.99% of the tweets. It is also notable that, although these references are not in the 40 most common tokens, #BlackLivesMatter tweets often come with specific allusions to well known victims of police violence such as George Floyd (2.58%) and Breonna Taylor (1.69%). COVID-19, which has heavily impacted the course of the protests and their public image, is mentioned in 2.18% of the tweets. Finally, Donald Trump comes up fairly often (2.64% just mentioned, 1.45% as a recipient of the tweet), as well as hashtags related to rejection of antiracism such as #AllLivesMatter (1.52%).

	Mean	Std Dev.	Q25	Q50	Q75
Length (in characters, before preproc.)	158.39	88.91	81	154	241
Length (in words, after preproc.)	10.97	6.56	5	10	16
Number of tweets by day	7253.00	111.41	7229	7279	7320
Number of replies	1.10	58.29	0	0	0
Number of favorites	18.01	945.05	0	0	2
Number of retweets	5.89	337.21	0	0	1
Number of involved users	1.23	0.82	1	1	1

Table 8.1: Description of extracted tweets characteristics

	Word	Frequency (%)		Word	Frequency (%)
1	black	15.98	21	today	3.17
2	people	10.78	22	say	3.13
3	#blm	9.06	23	time	3.12
4	lives	6.97	24	right	3.09
5	police	6.1	25	let	2.77
6	matter	5.97	26	want	2.69
7	white	5.82	27	trump	2.64
8	like	4.89	28	#georgefloyd	2.58
9	support	4.11	29	make	2.57
10	racism	4.06	30	protest	2.57
11	one	4.05	31	stop	2.46
12	get	3.95	32	would	2.44
13	racist	3.83	33	via	2.4
14	movement	3.48	34	change	2.39
15	know	3.47	35	love	2.38
16	need	3.45	36	day	2.33
17	see	3.34	37	going	2.29
18	justice	3.27	38	man	2.22
19	please	3.23	39	#covid	2.18
20	still	3.19	40	think	2.17

Table 8.2: Most frequent words in the corpus

8.2 General comparison

We fitted LDA, NMF, BTM and PTM for $K \in \{4, 16, 64\}$ on these data. For shorthand, we will write LDA-16 for an LDA model computed with $K = 16$, NMF-4 for an NMF model with $K = 4$, and so on. For all iterative computations, we set the maximum number of iterations at 200. This is important, as the default values in many implementations are too low and do not allow to reasonably assume convergence, especially with such a large corpus.

Before going into more details about specific models, we provide a general overview of the obtained results in Table 8.3. The table provides, for each combination of model and chosen K, the three metrics introduced in Section 6.4.

For a more visual representation of this information, we also provide a matrix of grouped bar plots to visualize metrics for all models (see Fig. 8.1). We can see that the PP decreases as the number of topics increases for all models, which is expected. The higher the perplexity, the less explaining power the model has for the data, and increasing K increases the amount of information retained by the model. The simple LDA model seems to perform the best, and PTM the worst overall. NMF seems to react most strongly to the increase in K, but is remarkably in the same range as the other models despite being the only one non-probabilistic.

These results can overall seem surprising, as PTM and BTM are specifically tuned to address the issues caused by short documents such as tweets. The reason for the lower PP of LDA could be that, even with the shorter format of tweets, its generative model is still the most adequate to describe the corpus. Of course, it could also originate from the difference in computation. We set the number of iterations to 200 for all models, but this does not control for the internal design choices that can make convergence more or less likely for each implementation.

We can observe the same trend of better results for higher K on the plot for UC results (see Fig. 8.1, second row), albeit in an inverted fashion since a higher coherence is preferable. Surprisingly, LDA shows the worst performances along with BTM, while PTM has the highest UC for $K = 16$ and $K = 64$. However, we can see that we have far fewer differences between models, and that they all score in the same range for an identical K.

These results were not expected, as UC measures the plausibility of co-occurence between the top words of a model, which is what the BTM models directly. This may be due to several factors, including the quality of the BTM implementation we use. It may also be worth mentioning that, although we deemed our corpus of sufficient size, the experiments conducted by Yan, Guo, Lan, et al. (2013) involved more than 4 million tweets. Additionally, their experiment with a smaller corpus was evaluated in a different and less

Model	K	PP	UC	GC
LDA	4	2578.09	661.80	4.59
	16	1706.62	747.02	4.90
	64	**1062.46**	876.06	4.26
NMF	4	3714.26	666.22	**6.75**
	16	2584.41	801.19	5.24
	64	1402.12	901.75	5.29
BTM	4	2759.79	665.59	4.42
	16	2010.30	786.61	4.33
	64	1377.61	864.08	4.23
PTM	4	3220.41	634.47	5.24
	16	2198.79	823.03	3.86
	64	1729.75	**926.18**	4.00

Table 8.3: Results for all evaluated models, with the best performance in bold for each metric

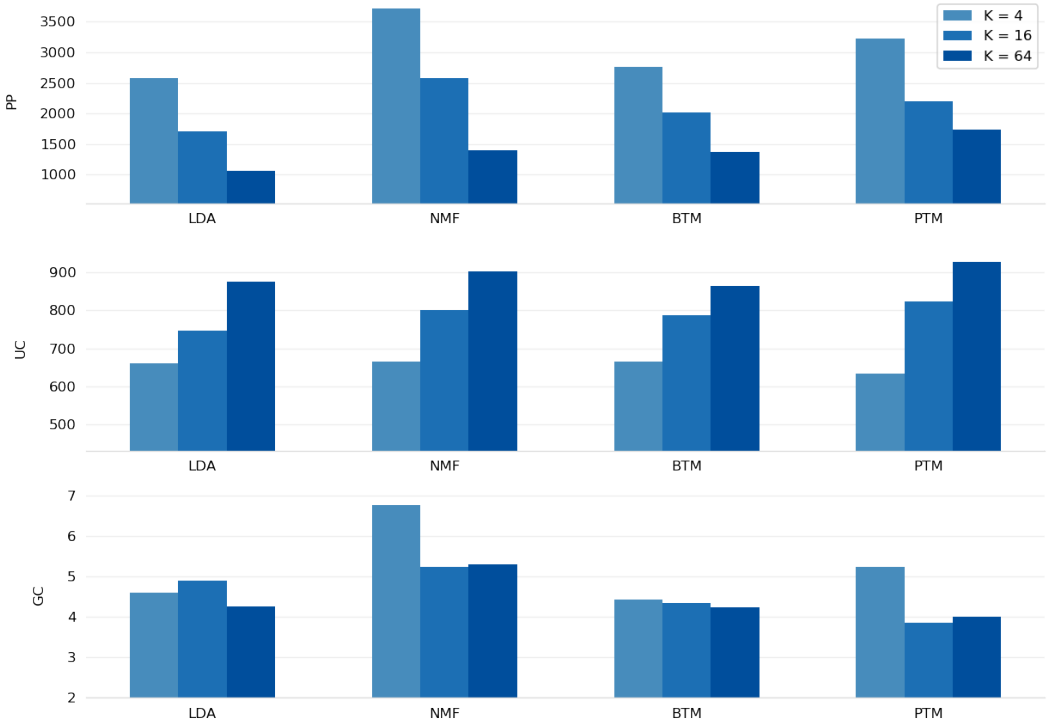

Figure 8.1: Comparison of four trained models, i.e. LDA, NMF, BTM and PTM (columns), for $K \in \{4, 16, 64\}$ (colors), on three metrics, i.e. PP, UC and GC (rows)

informative way. It may be the case the BTM performs better for very large corpora, where there are more documents to establish stable co-occurence patterns between words.

Finally, regarding GC, we see less of an association between K and performance (see Fig. 8.1) but it is not surprising, considering the lesser stability of this metric. We can nevertheless see that NMF has superior scores once again, with PTM taking the last position. NMF seems to have an abnormally high score for $K = 4$, but this should be examined further as it could easily be due to random noise. Indeed, this more experimental metric can be unstable, and in the case where $K = 4$ it is only averaged across four different Google searches.

8.3 Relations between metrics

Because the choice of evaluation metrics is of drastic importance to the conclusions we can make, we will now briefly discuss their relations to each other without consideration for the models. It is also interesting to check whether the findings of Chang et al. (2009), showing a negative relation between perplexity and coherence, can be reproduced here.

Model level First, we computed correlations at the model level, which means that we have only 12 data points for each metric. This is a small sample, but it still allows us to check correlations between PP and the other two metrics. The Spearman correlation coefficient of PP and UC is equal to $-.741$ ($p = .006$), and that of PP and GC is equal to .455 ($p = .137$). Contrary to expectations, we do observe a negative association between PP and UC. We observe, however, a positive relationship between PP and GC, albeit weaker and not statistically significant. Additionally, this coefficient is influenced by a single outlier, NMF-4 that has both the highest PP and the highest GC of all models (see Table 8.3), which drives up the correlation.

Topic level In order to compare UC and GC, on the other hand, one can rely on the individual topic scores across all models. This constitutes a sample of size $4 \times (4+16+64) = 336$. A representation of these data can be found in Fig. 8.2, along with a local polynomial regression line for easier visualization of the trend.

We can observe a clear negative relationship between UC and GC, and indeed their Spearman coefficient is equal to $-.582$ ($p < .001$). This is an interesting result which confirms our intuition that two coherence metrics can measure entirely different aspects of a topic. In our context, this may be due to a negative relation between words co-occuring in tweets and co-occuring in the web pages indexed by Google. This would not lend any new credibility to the GC metric, but it would indicate that tweets, perhaps specifically tweets mentioning #BlackLivesMatter, have their own co-occurence patterns that differ vastly from those of other online documents.

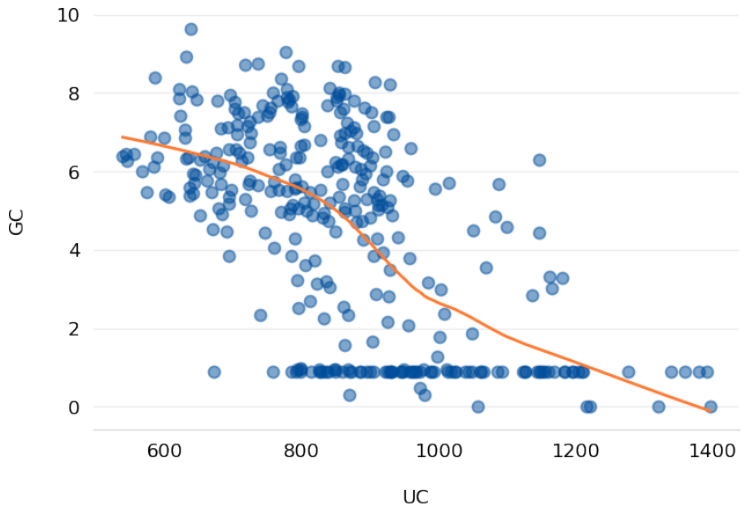

Figure 8.2: Relationship between UC and GC across all topics, independently of model or number of topics

Finally, we can see there are many GC scores concentrated at precisely 0.903, appearing in Fig. 8.2 as a straight line. This is due to Google claiming, for some reason, to find only 8 results for a given query. This occurs especially when the search engine only finds a few very related news articles. It is nonetheless interesting to see that those low GC scores occur almost exclusively for values of UC greater than approximately 800.

8.4 Model details

We will now focus on the topics obtained using our different models. For the sake of clarity, we have to refrain from going over all topics of all models, which are already summarized in Section 8.2. Instead, we will present a few topics from four models of our choice, based on the information we presented until now. In this section, we will especially try to relate the metrics of each topic with the subjective coherence of its top words. In order to describe a specific topic, we will denote it using z_k with $k \in \{1, \ldots, K\}$ as was done before. The value of k that is attributed to a topic has no bearing to its score or its rank among all topics.

8.4.1 LDA-64

We will first look at the LDA model we computed with 64 topics. As shown in Fig. 8.1, this model scores the lowest on PP, but performs worse than PTM and NMF on UC. As there are way too many topics to examine, we picked the two scoring the highest on UC (z_{20} and z_{30}), the two scoring the highest on GC (z_6 and z_{55}), as well as two random ones

	Top words	UC	GC
z_{20}	#seattleprotests seattle #seattle #chop #seattleprotest #blm #chaz #defundspd zone #premierleague @mayorjenny hill #pl #sufc #sheffieldunited #twitterblades #sheffield #blades autonomous #defundpolice	1203.753	0.903
z_{30}	#georgefloyd king #policereformnow #traitortrump martin #trumppandemicfailure #supremeloser #trumprussia luther #buildbackbetter #voteouthate #noborderwall #republicansvirus #nativeamerican #breonnataylor trump #blackjoy #wewantjoe #protectdemocracy #boycottgoya	1168.429	0.903
z_{6}	years ago time still last one today never would like since ever days weeks many day seen first long two	776.177	9.033
z_{55}	keep let change going people still need make stay get stop right fighting fight time safe see world everyone back	695.329	7.962
z_{35}	love people world change one see heart better hate hope like much need life black make way beautiful live children	702.180	7.777
z_{59}	via sign petition justice @change elijah mcclain county police martin attorney trayvon stop court district department rice law death #elijahmcclain	956.667	2.079

Table 8.4: Top 20 words of some topics found by LDA-64

(z_{35} and z_{59}). The top words of these topics, as well as their UC and GC scores, can be found in Table 8.4.

A very clear trend can be observed, where topics performing well on GC do not contain hashtags or mentions. This confirms our intuition that the GC metric is biased against Twitter-specific language. Conversely, topics ranking very high on UC seem to contain almost exclusively hashtags, which is more surprising. We could take it to mean that the strongest relations between words, in this corpus, are those between hashtags.

If we look at the top words, we notice at first glance that these topics look quite varied. This is expected, as here there are 64 different topics to balance the information of the corpus between. Topic z_{20}, the LDA topic scoring the highest on UC, seems to mix two elements together: the Capitol Hill Organized Protest (CHOP), sometimes called the Capitol Hill Autonomous Zone (CHAZ), an area of Capitol Hill that was barricaded and declared police-free, as well as European soccer competitions. This is probably related to professional athletes kneeling during anthems, a sign of protest that became widespread during the worldwide unrest of June 2020. Topic z_{30} seems to also be meaningful, with

	Top words	UC	GC
z_1	black white man women community men support lives killed woman crime person americans trans communities police racism care america owned	726.736	6.981
z_2	people white like racist know say need get think racism many see right stop want color still saying one hate	621.636	7.859
z_3	lives matter say saying #black movement still black #allivesmatter doesn blue mean said think matters believe really blm always life	716.733	5.290
z_4	#blm police like please support one get racism justice need racist know movement see time let today still make want	599.781	6.857

Table 8.5: Top 20 words of some topics found by NMF-4

terms centering around Trump himself and the presidential campaign that was unfolding at the time. Interestingly, those two topics contain indicators of two different trends in antiracist discourse: a revolutionary leftist one for z_{20}, centering the issue around autonomy and police funding, and a more mainstream liberal one for z_{30}, centering the issue around political candidates.

On the other hand, the topics selected for their high GC do not appear very interpretable, as they contain very generic words. The top words of the random topics, finally, seem more coherent. Topic z_{35} may be a bit vague, but contains positive words that could be linked with a more optimistic outlook on the issue, while z_{59} clearly centers around the case of Elijah McClain, a young african-american man who died in police custody in 2019. Moreover, we observe the same trend in which the GC metric appears inversely related with the specificity of a topic.

8.4.2 NMF-4

We chose to present this model because, as shown in Fig. 8.1, it scores very high on both UC (compared to other models with $K = 4$) and GC (compared to all models), while also exhibiting the worst results of all models on PP. In Table 8.5, we present the top words of each of its topic as well as their UC and GC scores.

Firstly, we notice that all GC scores are high, and we can conclude that the high GC average of this model was not due to some extreme outlier. The same is true for UC, which shows no extreme measure either. It also seems, at first glance, the words we find for each of these topics do not differ more between topics than they do inside topics. However, there are only three words that are shared among more than half of the topics: “men”, “racism”, and “still”. Furthermore, we do not really observe the same effect as for LDA-64, where

topic scoring higher on UC or GC would be more specific to tweets or not. This is likely due to having fewer topics, as we noticed upon further checks that the other three models also have fewer hashtags and mentions in their top words for $K = 4$.

Paying a closer look, we can see that in the top words of z_1 are words such as "community", "women", "trans", "owned" or "america", all potential indicators of leftist discourse with a high-level view of the issue and a focus on intersectionality. In comparison, we can see that the words of z_2 are more evocative of optimism and good intentions than radical policies. Topic z_3, on the other hand, has some words that are typical of right-wing reactions to the BLM movement. These are such as "#alllivesmatter", a frequent response misrepresenting the BLM statement as insisting only black lives matter, or "blue", a colloquialism for the police force used almost exclusively by its proponents. Topic z_4, finally, seems too generic to interpret.

8.4.3 BTM-16

We decided to examine results for BTM-16, which means there are again too many topics to go over each one of them. We will therefore settle again on examining the two topics with the highest UC (z_{10} and z_4), the two with the highest GC (z_{11} and z_9), and two random ones (z_2 and z_1). The top words of these topics can be found in Table 8.6, along with their UC and GC scores.

We observe, again, the same trend of topics with high UC having more hashtags and mentions, albeit more noticeably in z_{10} than in z_4. Topic z_{10} seems very similar to the z_{30} in LDA-64 (see Table 8.4). Indeed, we see different political slogans, as well as mentions of both Donald Trump and Joe Biden themselves. Topic z_4 seems straightforward to interpret, with multiple words seemingly referring to professional athletes kneeling during the national anthem as a sign of protest against systemic racism. Interestingly, this topic also has an acceptable GC score, contrary to z_{10}, in addition to being interpretable and having a high UC. We notice once again that the high-GC topics are especially vague, although one could make the case that z_{11} is linked with tributes to John Lewis, a congressman and a highly regarded civil rights figure who died in July 2020. Finally, the two random topics both seem to be interpretable. Indeed, z_2 shows top words related with the historical roots of systemic racism, such as "confederate", "statues" and "slavery". Topic z_1, on the other hand, seems to be specifically related with tweets about the victims of police violence, especially Breonna Taylor.

8.4.4 PTM-16

Finally, we will examine the top words of the topics found by PTM-16, as it has the highest results on UC of all the $K = 16$ models despite not performing well on PP and GC (see Fig. 8.1). We will use the same scheme as for other models with too many topics, and

	Top words	UC	GC
z_{10}	#georgefloyd @cnn #policereformnow @msnbc #traitortrump #trumppandemicfailure #trumprussia #supremeloser @abc #noborderwall #buildbackbetter @cbsnews #voteouthate #republicansvirus trump @washingtonpost @realdonaldtrump @joebiden #nativeamerican @nytimes	1155.154	0.903
z_4	players support knee team game anthem fans national kneeling #nba sports hockey @nfl league kneel nascar take movement nba see	1051.129	4.494
z_{11}	day today love one world life time john thank lewis america rights justice years people never great every must black	742.280	7.682
z_9	people like get black know one white still let going see #blm need say time racist would make right keep	578.783	6.892
z_2	black racism history via racist white slave slavery statue people american african statues confederate #blm #racism anti slaves years islam	746.661	4.442
z_1	justice police sign breonna #breonnataylor taylor petition via arrest killed cops #justiceforbreonnataylor still murdered floyd george murder officers #sayhername #blm	782.121	4.926

Table 8.6: Top 20 words of some topics found by BTM-16

	Top words	UC	GC
z_{11}	ding food #wwg #justiceforall united states paso jim first marijuana wga becomes crow city #pennstate outlaw #dcprotests #blacklivesmatterdc #cannabis #dcprotest	1159.958	0.903
z_{10}	#policereformnow #georgefloyd #traitortrump #voteouthate #trumppandemicfailure #supremeloser #trumprussia #buildbackbetter #noborderwall #americaortrump #wearadamnmask #republicansvirus trump #nativeamerican king #blm #blackjoy dollar #boycottgoya #wewantjoe	1143.231	0.903
z_9	day vote president john lewis god every name look got year one know racist party today rest house time slave	850.903	7.795
z_8	people police racist white like know right would want get fuck cops need think see even stop real still time	623.266	7.430
z_4	#breonnataylor breonna #justiceforbreonnataylor taylor justice arrest #sayhername cops still killed #georgefloyd #blm murdered #nojusticenopeace #elijahmcclain police #justiceforgeorgefloyd people murder arrested	793.667	0.954
z_5	protest city today sign justice street mural march new love racism front park #blm join last listening hate get outside	792.664	6.348

Table 8.7: Top 20 words of some topics found by PTM-16

will present the two topics with the highest UC (z_{11} and z_{10}), the two with the highest GC (z_9 and z_8), and two random ones (z_4 and z_5). The top words of these topics can be found in Table 8.7, along with their UC and GC scores.

Yet again, we observe that the topics that are scoring the highest on UC have their GC scores stuck at 0.903, and present a lot of hashtags. We notice that z_{10} groups several hashtags related to liberal critics of Donald Trump personally. It closely resembles z_{30} of LDA-64 (see Table 8.4), and z_{10} of BTM-16 (see Table 8.6). Topic z_{11}, on the other hand, seems a lot less focused, with two elements related to cannabis, another important political conversation in the USA, but not overall interpretable. Topics z_9 and z_8 seem not very interpretable either, although we can see terms related to John Lewis, religion and slavery in z_9. Topic z_4 seems to be about specific victims of police brutality, such as Elijah McClain and Breonna Taylor, and appears related with topics discussed earlier such as z_{59} in LDA-64 (see Table 8.4) and z_1 in BTM-16 (see Table 8.6). Finally, z_5 seems to be specifically about outside protest activities, with words like "protest", "march" and "street", but also "mural" and "park". This is a good instance of a topic that is coherent without

losing specificity.

Chapter 9

Discussion

The present thesis aimed to offer an overview of topic modeling literature, especially that which is pertaining to the modeling of short documents, and to compare different solutions that were proposed to address this specific issue. After carefully selecting models, we fitted them on both simulated and real-world data. We then proceeded to compare them on evaluation metrics we chose and implemented. We will now endeavor to review the findings of these comparisons, to present some limits of our work and, finally, to offer some perspectives on what could be the focus of future work in this area.

9.1 Findings

Our simulation study yielded interesting results, both expected and surprising. First and foremost, we observed a clear difference between the best models for short and long documents. BTM performed noticeably better on the former, while NMF had the best scores on the latter. LDA had performances almost on par with NMF, and PTM was the worst performing model in all settings.

Regarding the effect of our simulation settings, we did not notice any effect of the amount of noise we added except a slight negative one. This could indicate that adding more random words is a good idea, producing more realistic corpora, or that other methods to add noise should be investigated. We also noticed an effect of the true K, which interestingly was positive on UC but not PP.

Using real-word short documents, however, we obtained results that were somewhat more mixed. LDA performed the best when compared on PP, followed closely by BTM. On the other hand, the best model appeared to be PTM even though NMF was very close. Finally, it was NMF that obtained the highest GC scores. Regarding the number of topics, it appeared that the larger was always the best for all models and all metrics except for GC, where the lowest K was on average the better one.

Given the human aspect of the data, we also examined the top words of the topics we found with each of our models. First and foremost, we observed that, across all models, topics with a higher UC tended to have more vocabulary specific to tweets, whereas topics with a higher GC tended to have more general vocabulary. We also interpreted some topics based on their top words, and found it was possible to recognize well known trends in discourse from them. We also observed that the best topics for interpretability tended to have good scores on both metrics, but were not necessarily the best on any particular one. Still regarding interpretability of topics, it seemed that BTM fared particularly well, with NMF and LDA produced very interpretable topics as well. PTM seemed to perform slightly worse than the others but still produced interpretable topics. Finally, it seemed that the optimal number of topics for interpretability was dependent on the task. Fewer topics seemed convenient to have very broad trends, and larger numbers of topics to have more specific ones. For the purpose of interpreting models and in the context of our corpus, it appeared that 16 topics was a good middle ground.

We also examined the relation between PP, UC and GC on both simulated and real-world data. In both cases, we found PP and UC to be negatively related, which means that a lower perplexity goes with a higher UC. GC, which could only be used on real-word data, was found to have a negative relation with PP as well, albeit in a weaker fashion. Finally, UC and GC appeared to have strong negative relation in our context. This is interesting, as it shows how two metrics considered very similar can in this context favor very different topics.

9.2 Limits

One of the main limits we encountered was a lack of computational power. Indeed, the sacrifices we made in order to stay inside a reasonable timeframe had a pervasive influence on all our results. This is particularly true regarding the assumptions we usually make about convergence. In the real-world setting, we only had to compute our models once and could therefore set the number of iterations high enough to assume convergence. In the simulated setting, on the other hand, we had to set it considerably lower in order to make the experiment possible. This could have negatively and selectively affected the results of the models, as the implementations we use may need very different numbers of iterations to converge.

There are two main factors that affected negatively the required computation time. The first one is the BTM implementation, which is a standard Gibbs sampling algorithm with no optimization for parallel computing which means that, of all the models, it always took the longest time to train. Another implementation may be useful, especially considering its good performances on both simulated and real-world data. The second factor is the evaluation process. We implemented our metrics from scratch, but did not optimize this

code for parallel computing. Moreover, the process of computing GC sometimes required waiting periods.

One of the consequences of our limited resources is also the number of parameters we were able to investigate. Parameters of interest beyond K could have been the priors for the document-topic and the topic-words distributions, for instance. The fact that we did not vary J, the number of pseudo-documents used for PTM, may also have unfairly penalized this model. That being said, other considerations than computational power alone were responsible for reducing the scope of this work. Some of those are the inclusion of NMF, a method which has very few parameters in common with the others, and a deliberate choice to not take too much focus away from the comparison between models. Those are not, however, design choices we regret.

Nevertheless, considering the very good results of BTM on short documents and of NMF on documents of all length, it seems unfortunate that we did not find any reliable implementation of SMNF in Python. Indeed, it follows the same intuition as BTM and was reported to have good performances on short documents (Yan, Guo, Liu, et al., 2013). On the same note, the findings of Ding et al. (2008) and Z. Yang et al. (2011) are very relevant to the use of NMF as a topic modeling technique on text data. In hindsight, it would have been perhaps more interesting to normalize the data before factorizing it using KLD as a loss function, which is more efficient and allows better comparison with probabilistic models (Z. Yang et al., 2011). Instead, we had to resort to normalizing the components a posteriori, as was originally suggested by Ding et al. (2008). Another limiting factor is that we did not, contrary to their proposal, use GKLD as a loss function.

Finally, another limit of our work is the exclusive use of tweets as an example of short text. On the one hand, these data are used a lot in the literature and is therefore useful to compare our results. On the other hand, it would have been interesting to see which of our conclusions held on different real-world short documents, such as instant messages or news articles comments.

9.3 Perspectives

Finally, there are multiple directions in which the present work could be extended. One of our finding is that topics who score extremely high on one metric are not necessarily the best ones for human interpretability. In the field of topic model evaluation, it could therefore be interesting to design new metrics that take interpretability as a state of equilibrium and not as a value to maximize. Indeed, one could see the interpretability of a topic as being in a multidimensional space defined by different metrics, and its optimum as lying somewhere in the middle. Such a metric could, for instance, range from high specificity of topics to high generalizing power.

Additionally, even though we briefly discussed the subject, this work did not include DR in its scope. This was intentional, as this is a very broad topic involving a lot of different knowledge, but it is nevertheless very relevant to topic modeling for short documents. This is especially true because, as we noticed, there seems to be a negative relationship between internal and external coherence of topics. Using DR trained on an external corpus to compute topic models could bridge that gap, as well as help reduce the sparsity of short documents. In doing so, one would lose the information of documents, but as is shown by the good results of BTM this is not always needed.

Finally, an aspect of major importance on which we did not focus are the practical uses of topic modeling. Indeed, the models we use have wide ranges of application well beyond exploratory analysis of corpora. For instance, in real-life scenarios, topics are often used as features in predictive models for classification or regression tasks, which is a great way of reducing the dimensionality of a dataset and gaining computational efficiency without loosing too much information. Moreover, in most practical cases a classification task will require observations to be labeled, and it is frequent that only few of them are. In this context, a topic model fitted on all available data can help create features for labeled observations that contain information from the unlabeled data.

Beyond further use in statistical analysis, topic models can have real implications for people. Dimensionality reduction techniques are very important in recommender systems, an area of data science that is used and abused by social media algorithms, online shopping websites and other streaming media providers. However, in our opinion, these techniques are not put to such good use when it comes to helping people organize the information they receive. Topic modeling, specifically, could help people navigate large unstructured collections of documents. Therefore, we believe that a major area of extension could be the development of visualization tools that provide deep, rich insight about short documents in an intuitive manner.

Conclusion

The purpose of this thesis was to investigate the specific issues faced by topic modeling when dealing with shorter documents, which are becoming ubiquitous due to the generalized use of internet. First, we reviewed some aspects of literature relevant to natural language processing and to topic modeling specifically. It was decided to use LDA and NMF as standard topic modeling techniques for further comparisons. We then presented models that were specifically designed to address short documents. We retained BTM, modeling word co-occurences instead of documents, and PTM, aggregating documents before modeling them, to compare in our experiments.

We trained those models for different numbers of topics on multiple simulated corpora, as well as on one real-world corpus. The simulations were obtained using the generative process of LDA, with different settings across which we varied the length of the documents, the true number of topics and the amount of noise words added. We used three metrics, PP, UC and GC measuring different aspects of model fit and coherence, of which only the first two were useable on the simulated data.

On simulated data, we found BTM to be the model best suited to short documents across all settings, while NMF also performed very well. On real-world data, the results were more split, with LDA being the best model according to PP, PTM according to UC and NMF according to GC. Following a more subjective examination of the extracted topics, we found the topics of BTM to be the most interpretable. Overall, we were able to recognize real themes of the corpus among the extracted topics. We also noted that, while UC and PP were negatively associated, the relation between UC and GC was positive. We discussed these findings in detail, while also presenting some limits and perspectives for future work.

We believe that the main contribution of this work was to evaluate publicly available implementations of topic models, both parametric and non-parametric, in different settings, on different data, and on different metrics. Our findings support that at least one model designed for short documents is indeed best suited for this task, while also indicating that some non-parametric methods could be strong choices as well. We also noted that future work could focus on finding metrics based on equilibrium rather than optimum. This work is, nevertheless, not complete, as it does not include several methods that could have shown

good results. In particular, we have seen good results for a model built on the view of the corpus as word-populated graph. This seems a very important trail to follow in future research. As the time passes, we all become more connected; it would only make sense that our words and hashtags start making connections of their own.

Bibliography

Asuncion, A., Welling, M., Smyth, P., & Teh, Y. W. (2009). On Smoothing and Inference for Topic Models. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 27–34. https://doi.org/10.5555/1795114.1795118

bab2min, Fenstermacher, D., & Schneider, J. (2021). *Tomotopy* (Version v0.12.1). https://doi.org/10.5281/zenodo.5000206

Baeza-Yates, R., & Ribeiro, B. d. A. N. (2011). *Modern information retrieval*. New York : ACM Press ; Harlow, England : Addison-Wesley, retrieved February 14, 2020, from http://repository.fue.edu.eg/xmlui/handle/123456789/3692

Berry, M. W., Browne, M., Langville, A. N., Pauca, V. P., & Plemmons, R. J. (2007). Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, *52*(1), 155–173. https://doi.org/10.1016/j.csda.2006.11.006

Bicalho, P., Pita, M., Pedrosa, G., Lacerda, A., & Pappa, G. L. (2017). A general framework to expand short text for topic modeling. *Information Sciences*, *100*(393), 66–81. https://doi.org/10.1016/j.ins.2017.02.007

Blei, D. M., & Lafferty, J. D. (2006). Dynamic topic models. *Proceedings of the 23rd International Conference on Machine Learning*, 113–120. https://doi.org/10.1145/1143844.1143859

Blei, D. M., & Lafferty, J. D. (2007). A correlated topic model of Science. *The Annals of Applied Statistics*, *1*(1), 17–35. https://doi.org/10.1214/07-AOAS114

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022. https://doi.org/10.5555/944919.944937

Bouma, G. (2009). Normalized (Pointwise) Mutual Information in Collocation Extraction. *Proceedings of the Biennial GSCL Conference 2009*.

Catral, M., Han, L., Neumann, M., & Plemmons, R. J. (2004). On reduced rank nonnegative matrix factorization for symmetric nonnegative matrices. *Linear Algebra and its Applications*, *393*, 107–126. https://doi.org/10.1016/j.laa.2003.11.024

Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., & Blei, D. M. (2009). Reading tea leaves: How humans interpret topic models. *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, 288–296.

Chen, X., Hu, X., Shen, X., & Rosen, G. (2010). Probabilistic topic modeling for genomic data interpretation. *2010 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 149–152. https://doi.org/10.1109/BIBM.2010.5706554

Chung, M.-H., Wang, Y., Tang, H., Zou, W., Basinger, J., Xu, X., & Tong, W. (2015). Asymmetric author-topic model for knowledge discovering of big data in toxicogenomics. *Frontiers in Pharmacology*, *6*. https://doi.org/10.3389/fphar.2015.00081

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407. https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, *39*(1), 1–22. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x

Ding, C., He, X., & Simon, H. D. (2005). On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering. *Proceedings of the 2005 SIAM International Conference on Data Mining*, 606–610. Retrieved March 14, 2021, from https://epubs.siam.org/doi/abs/10.1137/1.9781611972757.70

Ding, C., Li, T., & Peng, W. (2008). On the equivalence between Non-negative Matrix Factorization and Probabilistic Latent Semantic Indexing. *Computational Statistics & Data Analysis*, *52*(8), 3913–3927. https://doi.org/10.1016/j.csda.2008.01.011

Doyle, G., & Elkan, C. (2009). Accounting for Burstiness in Topic Models. *Proceedings of the 26th Annual International Conference on Machine Learning*, 281–288. https://doi.org/10.1145/1553374.1553410

Du, R., Kuang, D., Drake, B., & Park, H. (2017). Hierarchical community detection via rank-2 symmetric nonnegative matrix factorization. *Computational Social Networks*, *4*(1), 7. https://doi.org/10.1186/s40649-017-0043-5

Erosheva, E., Fienberg, S., & Lafferty, J. (2004). Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences*, *101*, 5220–5227. https://doi.org/10.1073/pnas.0307760101

Fei-Fei, L., & Perona, P. (2005). A Bayesian hierarchical model for learning natural scene categories. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, *2*, 524–531 vol. 2. https://doi.org/10.1109/CVPR.2005.16

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Gao, W., Peng, M., Wang, H., Zhang, Y., Xie, Q., & Tian, G. (2019). Incorporating word embeddings into topic modeling of short text. *Knowledge and Information Systems*, *61*(2), 1123–1145. https://doi.org/10.1007/s10115-018-1314-7

Gonzalez, E. F., & Zhang, Y. (2005). Accelerating the Lee-Seung Algorithm for Nonnegative Matrix Factorization. Retrieved August 5, 2020, from https://scholarship.rice.edu/handle/1911/102034

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, *101*, 5228–5235. https://doi.org/10.1073/pnas.0307752101

Griffiths, T. L., Steyvers, M., Blei, D. M., & Tenenbaum, J. B. (2004). Integrating Topics and Syntax. *Proceedings of the 17th International Conference on Neural Information*

Processing Systems, 537–544. Retrieved October 16, 2019, from http://dl.acm.org/citation.cfm?id=2976040.2976108

Hoffman, M., Bach, F. R., & Blei, D. M. (2010). Online Learning for Latent Dirichlet Allocation (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, & A. Culotta, Eds.). *Advances in Neural Information Processing Systems*, *23*, 856–864. Retrieved October 16, 2019, from http://papers.nips.cc/paper/3902-online-learning-for-latent-dirichlet-allocation.pdf

Hofmann, T. (1999). Probabilistic Latent Semantic Analysis. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 289–296. Retrieved October 17, 2019, from http://dl.acm.org/citation.cfm?id=2073796.2073829

Hofmann, T., Puzicha, J., & Jordan, M. I. (1998). Learning from dyadic data. *Proceedings of the 11th International Conference on Neural Information Processing Systems*, 466–472.

Hong, L., & Davison, B. D. (2010). Empirical study of topic modeling in Twitter. *Proceedings of the First Workshop on Social Media Analytics*, 80–88. https://doi.org/10.1145/1964858.1964870

Jiang, S., Qian, X., Shen, J., Fu, Y., & Mei, T. (2015). Author Topic Model-Based Collaborative Filtering for Personalized POI Recommendations. *IEEE Transactions on Multimedia*, *17*(6), 907–918. https://doi.org/10.1109/TMM.2015.2417506

Jipeng, Q., Zhenyu, Q., Yun, L., Yunhao, Y., & Xindong, W. (2019). *Short Text Topic Modeling Techniques, Applications, and Performance: A Survey*. arXiv: 1904.07695 [cs]. Retrieved April 17, 2020, from http://arxiv.org/abs/1904.07695

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. (1999). An Introduction to Variational Methods for Graphical Models. *Machine Learning*, *37*(2), 183–233. https://doi.org/10.1023/A:1007665907178

Kuang, D., Ding, C., & Park, H. (2012). Symmetric Nonnegative Matrix Factorization for Graph Clustering. *Proceedings of the 2012 SIAM International Conference on Data Mining*, 106–117. Retrieved August 10, 2020, from https://epubs.siam.org/doi/abs/10.1137/1.9781611972825.10

Kuang, D., & Park, H. (2013). Fast rank-2 nonnegative matrix factorization for hierarchical document clustering. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 739–747. https://doi.org/10.1145/2487575.2487606

Lau, J. H., Newman, D., & Baldwin, T. (2014). Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Lau, J. H., Newman, D., Karimi, S., & Baldwin, T. (2010). Best Topic Word Selection for Topic Labelling. *Coling 2010: Posters*, 605–613. Retrieved July 21, 2020, from https://www.aclweb.org/anthology/C10-2069

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788–791. https://doi.org/10.1038/44565

Lee, D. D., & Seung, H. S. (2001). Algorithms for Non-negative Matrix Factorization. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in Neural Informa-*

tion Processing Systems 13 (pp. 556–562). MIT Press. Retrieved August 3, 2020, from http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

Li, X., Gan, J. Q., & Wang, H. (2018). Collective sparse symmetric non-negative matrix factorization for identifying overlapping communities in resting-state brain functional networks. *NeuroImage*, *166*, 259–275. https://doi.org/10.1016/j.neuroimage.2017.11.003

Mehrotra, R., Sanner, S., Buntine, W., & Xie, L. (2013). Improving LDA topic models for microblogs via tweet pooling and automatic labeling. *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 889–892. https://doi.org/10.1145/2484028.2484166

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., & McCallum, A. (2011). Optimizing semantic coherence in topic models. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 262–272.

Mottl, D. (2021). *GetOldTweets3*. Retrieved August 12, 2021, from https://github.com/Mottl/GetOldTweets3

Newman, D., Asuncion, A., Smyth, P., & Welling, M. (2009). Distributed algorithms for topic models. *Journal of Machine Learning Research*, *10*(8), 1801–1828.

Newman, D., Karimi, S., & Cavedon, L. (2009). External evaluation of topic models. *Proceedings of the Fourteenth Australasian Document Computing Symposium (ADCS 2009)*, 11–18.

Newman, D., Lau, J. H., Grieser, K., & Baldwin, T. (2010). Automatic evaluation of topic coherence. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 100–108.

Nguyen, D. Q., Billingsley, R., Du, L., & Johnson, M. (2015). Improving Topic Models with Latent Feature Word Representations. *Transactions of the Association for Computational Linguistics*, *3*, 299–313. https://doi.org/10.1162/tacl_a_00140

Nigam, K., Mccallum, A. K., Thrun, S., & Mitchell, T. (2000). Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, *39*(2), 103–134. https://doi.org/10.1023/A:1007692713085

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

Pedrosa, G., Pita, M., Bicalho, P., Lacerda, A., & Pappa, G. L. (2016). Topic Modeling for Short Texts with Co-occurrence Frequency-Based Expansion. *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, 277–282. https://doi.org/10.1109/BRACIS.2016.058

Quan, X., Kit, C., Ge, Y., & Pan, S. J. (2015). Short and sparse text topic modeling via self-aggregation. *Proceedings of the 24th International Conference on Artificial Intelligence*, 2270–2276.

Ramage, D., Dumais, S., & Liebling, D. (2010). Characterizing Microblogs with Topic Models. *Fourth International AAAI Conference on Weblogs and Social Media*. Retrieved July 25, 2020, from https://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/view/1528

Rosen-Zvi, M., Griffiths, T., Steyvers, M., & Smyth, P. (2004). The Author-topic Model for Authors and Documents. *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, 487–494. Retrieved October 16, 2019, from http://dl.acm.org/citation.cfm?id=1036843.1036902

Ruder, S. (2017). *An overview of gradient descent optimization algorithms*. arXiv: 1609.04747. Retrieved February 21, 2021, from http://arxiv.org/abs/1609.04747

Shi, T., Kang, K., Choo, J., & Reddy, C. K. (2018). Short-Text Topic Modeling via Non-negative Matrix Factorization Enriched with Local Word-Context Correlations. *Proceedings of the 2018 World Wide Web Conference*, 1105–1114. https://doi.org/10.1145/3178876.3186009

Shi, X., Lu, H., He, Y., & He, S. (2015). Community Detection in Social Network with Pairwisely Constrained Symmetric Non-Negative Matrix Factorization. *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 541–546. https://doi.org/10.1145/2808797.2809383

Steyvers, M., & Griffiths, T. L. (2007). Probabilistic topic models. In T. Landauer, D. McNamara, S. Dennis, & W. Kintsch (Eds.), *Latent Semantic Analysis: A Road to Meaning* (Laurence Erlbaum, pp. 424–440).

Taehoon, K., & Wurster, K. (2021, August 5). *Emoji*. Retrieved August 6, 2021, from https://github.com/carpedm20/emoji

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, *101*(476), 1566–1581.

Terpilowski, M. (2021, July). *Bitermplus* (Version v0.6.8). https://doi.org/10.5281/zenodo.5126579

Van Rossum, G., & Drake Jr., F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Wallach, H. M., Murray, I., Salakhutdinov, R., & Mimno, D. (2009). Evaluation methods for topic models. *Proceedings of the 26th Annual International Conference on Machine Learning*, 1105–1112. https://doi.org/10.1145/1553374.1553515

Wang, C., & Blei, D. M. (2009). Decoupling Sparsity and Smoothness in the Discrete Hierarchical Dirichlet Process. *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*, 1982–1989. Retrieved October 17, 2019, from http://dl.acm.org/citation.cfm?id=2984093.2984315

Wang, Y. (2008). Distributed Gibbs sampling of latent topic models: The gritty details.

Wang, Y.-X., & Zhang, Y.-J. (2013). Nonnegative Matrix Factorization: A Comprehensive Review. *IEEE Transactions on Knowledge and Data Engineering*, *25*(6), 1336–1353. https://doi.org/10.1109/TKDE.2012.51

Weng, J., Lim, E.-P., Jiang, J., & He, Q. (2010). TwitterRank: Finding topic-sensitive influential twitterers. *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, 261–270. https://doi.org/10.1145/1718487.1718520

Yan, X., Guo, J., Lan, Y., & Cheng, X. (2013). A biterm topic model for short texts. *Proceedings of the 22nd International Conference on World Wide Web*, 1445–1456. https://doi.org/10.1145/2488388.2488514

Yan, X., Guo, J., Liu, S., Cheng, X., & Wang, Y. (2013). Learning Topics in Short Texts by Non-negative Matrix Factorization on Term Correlation Matrix. *Proceedings of the 2013 SIAM International Conference on Data Mining*, 749–757. https://doi.org/10.1137/1.9781611972832.83

Yang, S., Lu, W., Yang, D., Yao, L., & Wei, B. (2015). Short Text Understanding by Leveraging Knowledge into Topic Model. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1232–1237. https://doi.org/10.3115/v1/N15-1131

Yang, Z., Zhang, H., Yuan, Z., & Oja, E. (2011). Kullback-Leibler Divergence for Non-negative Matrix Factorization. In T. Honkela, W. Duch, M. Girolami, & S. Kaski (Eds.), *Artificial Neural Networks and Machine Learning – ICANN 2011* (pp. 250–257). Springer. https://doi.org/10.1007/978-3-642-21735-7_31

Yin, J., & Wang, J. (2014). A Dirichlet multinomial mixture model-based approach for short text clustering. *KDD '14*. https://doi.org/10.1145/2623330.2623715

Zhao, W., Chen, J. J., Perkins, R., Liu, Z., Ge, W., Ding, Y., & Zou, W. (2015). A heuristic approach to determine an appropriate number of topics in topic modeling. *BMC Bioinformatics*, *16*(13), S8. https://doi.org/10.1186/1471-2105-16-S13-S8

Zuo, Y., Wu, J., Zhang, H., Lin, H., Wang, F., Xu, K., & Xiong, H. (2016). Topic Modeling of Short Texts: A Pseudo-Document View. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2105–2114. https://doi.org/10.1145/2939672.2939880

Appendices

A1 Notations

Numbers

V Number of possible words in a corpus

N Number of words in a document

M Number of documents in a corpus

K Number of topics in a corpus

L Number of possibles authors (specific to the ATM)

J Number of pseudo-documents (specific to the PTM)

I Number of biterms in a corpus (specific to the BTM)

R Length of distributed representations

ζ Average number of words in the documents of a corpus

$\tilde{K}$ Number of topics specified to compute a topic model

ν Multiplicative factor used when both K and $\tilde{K}$ are known, such that $\tilde{K} = \nu K$

Vectors

In parentheses, the lengths of the vectors.

w (V) Representation of a word

d (V) Representation of a document in vectorized form

z (K) Representation of a topic

a (L) Representation of an author (specific to the ATM)

p (J) Representation of a pseudo-document (specific to the PTM)

b (I) Representation of a biterm

α (K) Parameter for the distribution of topic according to the document

η (V) Parameter for the distribution of word according to the topic

λ (K) Variational parameter for the distribution of topics according to the document (specific to LDA)

γ (V) Variational parameter for the distribution of words according to the topic (specific to LDA)

ϕ (V) Variational parameter representing the probability of words according to the topic (specific to LDA)

ι (J) Parameter for the distribution of long documents according to the short document (specific to PTM)

Matrices

In parentheses, the numbers of rows and columns.

D $(N \times V)$ Representation of a document

C $(M \times V)$ Representation of a corpus, with documents in vectorized forms

A $(M \times L)$ Representation of the authors of the documents in a corpus (specific to the ATM)

B $(V \times V)$ Representation of the corpus as word co-occurences (specific to the BTM)

Z_w $(K \times V)$ Representation of the assignments of topics to words (specific to the BTM)

Z_b $(K \times I)$ Representation of the assignments of topics to biterms (specific to the BTM)

θ $(M \times K)$ Probabilities of topics according to the document (specific to LDA)

β $(K \times V)$ Probabilities of words according to the topic (specific to LDA)

κ $(M \times J)$ Probabilities of long documents according to the short document (specific to the PTM)

Σ $(K \times K)$ Diagonal matrix containing singular values (specific to LSI)

U_1 $(M \times K)$ Row singular vectors (specific to LSI)

U_2 $(K \times V)$ Columns singular vectors (specific to LSI)

W $(M \times K)$ Row components (specific to NMF)

H $(K \times V)$ Column components (specific to NMF)

Sets

$\mathcal{C}$ Representation of an unordered corpus

$\mathcal{W}$ Representation of the top words of a topic

Functions

$\mathbf{1}_{\{E\}}$ Indicator function, worth 1 if E is true and 0 otherwise

$\Psi(x)$ Digamma function

$D(a, b)$ Divergence between a and b

$P(E)$ Probability of E

$Q(E)$ Variational probability of E

$L(E)$ Log-likelihood of E

$F(w)$ The number of documents of the corpus in which a word w appears

A2 Acronyms

ATM Author-Topic Model

BLM Black Lives Matter

DR Distributed Representations

EM Expectation-Maximization

GC Google Coherence

GKLD Generalized Kullback-Leibler Divergence

IR Information Retrieval

KL Kullback-Leibler

LDA Latent Dirichlet Allocation

LSA Latent Semantic Analysis

MCMC Markov Chain Monte Carlo

MU Multiplicative Updates

NMF Non-negative Matrix Factorization

NPMI Normalized Pointwise Mutual Information

pLSA Probabilistic Latent Semantic Analysis

PMI Pointwise Mutual Information

PP Pseudo-Perplexity

PTM Pseudo-Document-Based Topic Modeling

SATM Self-Aggregation Based Topic Model

SED Squared Euclidean Distance

SNMF Symmetric Non-negative Matrix Factorization

SVD Singular Value Decomposition

TF Term Frequency

TF-IDF Term Frequency-Inverse Document Frequency

UC UMass Coherence

VI Variational Inference

It is hard to summarize information. This is especially true when the information is high-dimensional, such as text data. Topic models are tools to summarize this kind of data by assuming the existence of underlying abstract topics. Unfortunately, they face different issues when applied on short documents, like an increased sparsity.

In this thesis, we review existing topic models and compare different solutions that were proposed to address the short document issue. After carefully selecting models, we fit them on multiple simulated datasets, which we generate with various settings, and real-word data consisting of tweets related to the #BlackLivesMatter movement. The results of these comparisons are discussed, along with the limits of the experiments and perspectives for future work.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Faculté des sciences
Place des sciences, 2 bte L6.06.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/sc**