# Delivery Report

Progetto RASTA, fase di Topic Modeling

Contents

1. Topic Modeling 2

2. Word Representation 3

2.1. Local Representation 3

2.2. Distributional Embeddings 4

2.3. Contextual Embeddings 5

3. Document Representations 6

3.1. Bag of Words 6

3.1.1. Limitations 6

3.2. Distributed Representations of Sentences and Documents 7

3.3. Contextualized Document and Sentence Embeddings 7

4. Topic Models 8

4.1. Notation 8

4.2. Probabilistic Topic Models 9

4.2.1. Latent Dirichlet Allocation (LDA) 9

4.3. Hyperparameter Selection 13

4.3.1. Grid Search 13

4.3.2. Random Search 13

4.3.3. Bayesian Optimization 14

4.4. Neural Topic Modeling 15

4.4.1. Neural Variational Document Model (NVDM) 15

4.4.2. Embedded Topic Model (ETM) 15

4.4.3. Product of Experts LDA (PRODLDA) 16

## List of Figures

[**Figure 1**: Overview of topic modeling. A corpus of documents is given as input and the model returns the list of topics and the topic representations of the documents. 2](#_Toc174784587)

[**Figure 2**: An example of word vector representation generated from text. 4](#_Toc174784588)

[**Figure 3**: LDA in plate notation. The variable wnd, representing the n-th word of document d, is observed, then it is represented by a gray circle. While the other variables are unobserved, thus they are represented by white circles. Variables are repeated if they are included in a rectangle. 9](#_Toc174784589)

[**Figure 4**: Grid and random search of 9 trials for optimizing a function with 13](#_Toc174784590)

[**Figure 5**: Bayesian Optimization 14](#_Toc174784591)

[**Figure 6:** High-level schema of Neural Variational Document Model. 15](#_Toc174784592)

## List of Tables

[**Table 1**: Main notations for LDA and its extensions. 8](#_Toc172841220)

## Topic Modeling

Topic modeling is a statistical method for identifying the underlying themes or topics in a group of documents. Those models can reveal latent semantic structures in data by automatically detecting patterns of word co-occurrence. This makes possible to cluster documents or any kind of large textual collection based on specific statistical properties of the data.

 Topic models are a class of models that provide an automatic way to analyze the main themes of large volumes of texts. A topic model describes a corpus of documents through a set of fixed topics, where each topic is represented by its most significant words. Figure 1.1 sketches how a topic model works, along with its input and outputs.

**Figure 1**: Overview of topic modeling. A corpus of documents is given as input and the model returns the list of topics and the topic representations of the documents.

A topic model presents the information in a compact and interpretable form. For example, as seen in the picture, a topic characterized by the words “*learning, machine, deep, neural, network”* can be easily interpreted as a topic related to deep learning, or the words “*probability, distribution, gaussian, variable, random*” are related to probability theory. However, a topic is not just an unordered list of keywords: each word of the vocabulary has a specific weight, or probability weight, that identifies the importance of the word in the topic.

Not only does a topic model summarize a corpus by lists of coherent keywords, but each document can be described by the discovered topics in different proportions. Indeed, a document is rarely characterized by a single topic, rather it may talk about multiple topics. For example, an NLP paper can contain 30% of linguistics and 70% of computer science.

The two main elements in topic modeling are the documents and its constituents, i.e. the words. To allow the topic model to deal with these elements, we need to find a way to represent them under a computational point of view. This Chapter is therefore organized as follows: we will provide an overview of the main representation methods for words and documents in Section 1.2. and Section 1.3. respectively. In this way, we will have the fundamentals to provide some details on the focus of this thesis, i.e. probabilistic topic models, in Section 1.4. Since these models are usually controlled by hyperparameters, we will also provide an overview of the main methodologies for estimating the hyperparameters in topic models in Section 1.5.

## Word Representation

Language is made of words that under a computational point of view come from a vocabulary and we need to find ways to account for the meaning of these words under a computational point of view. Nowadays, the most common way to represent words in NLP is to use **vectors in a vector space**: words are embedded in a multi-dimensional vector space and we can therefore interpret them as points in a space that can be compared. The process of “embedding” words in the vector space is what brought the community to call these representations *word embeddings*. "Word embeddings" is in fact the general term that is used to refer to this kind of representations.

We can distinguish between two different ways of representing words with vectors: we refer to the first as a **local representation** and to the second one as a **distributed representations** (Ferrone, 2017). This distinction derives from one of the most vivid debates in the AI field during the ’80s on how to store conceptual information inside neural algorithms. Local representations are meant to represent a single concept with the activity of a single neural unit. On the other hand, distributed representations are meant to account for a pattern of activity of more neural units (Hinton, 1989).

### Local Representation

The simplest way to represent words in a way that is interpretable from a machine consists in using **one-hot encoding**. In one-hot encoding, each word is represented by a single and unique vector. One-hot encoding maps the i-th word of the vocabulary *V* to the vector in a vector space , where n is the cardinality of the vocabulary *V* and the i-th element of is set to 1, while all the other elements are set to zero. We generally refer to this kind of vectors as the *one-hot vectors*.

For example, given the words of the vocabulary V = {the, cat, is, on, table}, the word “*cat*” of the vocabulary *V* can be represented as a vectors of zeros with only one 1 in the position indexed by its own index in the vocabulary = <0, 1, 0, 0, 0> (i.e., “*cat*” is the second element and thus the 1 will be in the second component of the vector).

If we want to represent all the unique words in a text corpus, we will then need a matrix whose dimension is *V x V* (where *V* is the number of unique words). A one-hot encoded representation is simple but results in two main issues. First, the dimension of the matrix grows as the number of unique words increases. Encoding all the words of the English vocabulary would generate a matrix of at least 170, 000 x 170, 000 entries. Related to this issue, the resulting matrix is extremely sparse (each row is composed of all 0-valued entries except for one entry). Each vector is orthogonal to each other, therefore not representing any type of relationship between words. It is instead more convenient that word embeddings reflect and preserve certain specific properties of language. For example, we may agree that the word “*cat*" is more like "*dog*" than to “*Rome*". In the next section, we will see how this problem can be addressed by distributional representations.

### Distributional Embeddings

Distributional semantics is an approach to semantics that advocates a “usage-based” perspective on the computation of word meaning. Distributional semantics assumes that the statistical distribution and the frequency of usage of words inside textual documents can reveal information about the meaning of words themselves: words meaning can be found in the context (Lenci, 2008).

The definition of the concept "context" can vary widely across the different algorithms. The simplest case of context is co-occurrence: a word appears in the context of those words it co-occurs with. We expect the words “*cat*" and “*kitten*" to occur in similar contexts and thus being similar. Let us notice that also the words *cat* and *dog* could co-occur in some contexts, thus making the two words similar, but less similar than “*cat*" and “*kitten*" which co-occur more often. On the other hand, words that occur in different contexts, such as “*smartphone*", will not be similar to “*cat*". This effect allows us to define a graded similarity. In other words, the degree of semantic similarity between two words and is a function of the similarity of the contexts in which and usually appear. We in fact expect that the meaning of the words “*dog*" and “*cat*" to be similar, since both are domestic animals, have four legs, an owner, they eat, and so on.

Immagine che contiene linea, diagramma, schermata, testo

Descrizione generata automaticamenteModels that are based on distributional semantics aim to create representations in which similar vectors should represent similar words (i.e., words that occur in similar contexts). These algorithms take large amounts of text in input to create these vector representations. Figure 2.1 shows an example of what a vector space model built under the distributional hypothesis should create: “cats” and “dogs” are similar words and tend to occur in similar contexts (e.g., those shared by animals, those shared by house pets, etc...) and they tend to share fewer contexts with words like “president”.

**Figure 2**: An example of word vector representation generated from text.

There are different ways to generate these representations. One of the most famous models that can create distributional representations of words is **Word2vec** (Mikolov, 2013). Word2vec is a neural architecture that has been proposed in two different variants: Continuous Bag-of-words (CBOW) and Skip-gram (SG). Both architectures are simple feed-forward neural networks with one hidden layer, and they are trained over a large corpus of text. There are no non-linearities between the layers (except for a softmax function to compute the output scores of the network) and thus the projections are linear. The training examples for the models are extracted from text and are generally based on the concept of target word and context words that appear inside the corpus within a fixed distance from the target word: for example, in a sentence like “*the cat is on the table*”, the word “*cat*” might be the target word and “*the*”, “*is*”, “*on*”, “*the*”, “*table*” the context. CBOW gets the context words as input, and it aims at predicting the target words. Instead, SG is trained by considering the task in the opposite way: given the target word the model, it aims at predicting the surrounding words of the target. Once the models have been trained, the word embeddings are extracted from the first weight matrix of the neural network.

Word embeddings learned by Word2vec exhibited a good capability at capturing syntactic and semantic regularities in a language (Mikolov, 2013). In fact, the introduction of Word2vec represents a milestone for the NLP field. Different improved distributional embeddings models have been then proposed across the years (Grave, 2018) and have become ubiquitous in NLP (Khattak, 2019). However, these approaches have some limitations. Despite their capabilities of capturing syntactic and semantic regularities, it has been shown that these representations also capture bias in language (Caliskan, 2017). Moreover, most of these models also assign to each word a single vector representation, following that they compress all the senses of a word into a single vector.

### Contextual Embeddings

The word representations we have seen so far are just static representations of words: each word is associated with a single vector representation, regardless of the context. However, words change their meaning depending on the context in which they appear. Let us consider the following two sentences “*the Broadway play premiered yesterday*" and “*two teams play a football match*". The word “*play*" in the two sentences has two different meanings and syntactic roles.

Contextualized words embeddings aim at overcoming this issue and capturing word meaning in different contexts. We therefore aim to obtain two different vectors for the same word “*play*". In particular, we want to obtain a vector representation for the word “*play*" that is dependent on its context. Let be a document composed of , ,…, words. Then a context-dependent (or contextualized) vector representation for the k-th word of the document is

=

such that the representation changes for different contexts and f is function that maps the word to a continuous vector representation.

To obtain these vector representations we have to resort to the concept of language modeling. Language modeling is the task of predicting the next word given a sequence of words. For example, given the following sentence “*two teams play a football [BLANK]*”, a language model must predict the word in the *[BLANK]* position, which can be “*match*". It is intuitive that a language model is therefore required to be able to express syntax (the grammatical form of the predicted word must match its modifier or verb) and to model semantics.

More recent work, namely deep neural language models such as ELMo (Peters, 2018), BERT (Devlin, 2019) ,and GPT-2 (Radford A. W., 2019) , have successfully created contextualized word representations. Their internal representations of words are in fact called contextualized word representations because they are a function of the entire input sentence. The success of this approach suggests that these representations capture highly transferable and task-agnostic properties of language (Liu, 2019).

ELMo (Peters, 2018) creates contextualized representations of each token by concatenating the internal states of a bidirectional LSTM trained on a bidirectional language modeling task. On the other hand, BERT (Devlin, 2019) and GPT-2 (Radford A. W., 2019) are transformer-based language models (Vaswani, 2017). BERT is bidirectional like ELMo, while GPT-2 is a unidirectional language model. Each transformer layer of BERT and GPT-2 creates a contextualized representation of each token by attending to different parts of the input sentence. BERT – and subsequent iterations on BERT (Liu, 2019)– have achieved stateof- the-art performance on various downstream NLP tasks, ranging from question answering (Liu, 2019), natural language inference (Yang, 2019), and sentiment analysis (Yang, 2019).

## Document Representations

### Bag of Words

The Bag-of-Words (BoW) model is a document representation that turns text in natural language into a fixed-length vector. We can obtain the BoW representation of a document by first tokenizing the text, i.e. dividing the words or phrases in tokens. Then we can create a vector, whose length is equal to the number of unique tokens in the texts. The entries of this vector may be binary, thus indicating whether a token occurs (value 1) or not (value 0) in the considered document or can represent the counts of the tokens in the document. Let us consider the sentences "*The cat is on the table*" and "*The cat and the dog are under the table*". The vocabulary is composed of the 10 unique words: "The", "cat", "is", "on", "the", "table" "and, "dog", "are", "under". Notice that "The" and "the" are two different words. Considering the previous order of the words, the binary BoW representations of the two sentences are the following vectors: [1, 1, 1, 1, 1, 0, 0, 0, 0, 0] and [1, 1, 0, 1, 1, 1, 1, 1, 1, 1]. On the other hand, if we want to consider the counts, we will obtain the following vector representations: [1, 1, 1, 1, 1, 0, 0, 0, 0, 0] and [1, 1, 0, 1, 2, 1, 1, 1, 1, 1].

#### Limitations

Let us notice that the BoW model loses the contextual information of a document. the sentences “*the department chair couches offers*” and “*the chair department offers couches*” are represented by the same bag of words, but have different meanings. We do not know anymore in which position a given word appeared or which were their surroundings words. Nonetheless, this kind of representation can be useful from the point of view of the computational costs, but let’s look more closely at the specific problems:

* **Word association**. Bag of words makes the assumption that words in a document or corpus are unrelated to one another. Bag of words model extracts each word in a document as a feature, with term frequency serving as that feature’s weight.
* **Compound Terms**. Word correlation also applies to compound phrase representations in bags of words, when two or more words function as a single semantic unit. The encoding of multi-word concepts in a bag of words is inadequate in capturing their semantic and grammatical complexity.
* **Polysemy**. Numerous words have several distinct meanings. Bag of words combines all of these different meanings into a single word, eliminating potentially important information about the subject of a text (and hence potential classification), because it does not take context or meaning into account when modeling words.
* **Sparsity**. Every word in a bag of words model is a dimension, or feature, and every “document” is a vector. Several of the feature values for a particular vector may be zero since a document may not use every term in the produced model’s vocabulary. The model is said to be sparse (or, if vectors are represented as a matrix, a sparse matrix) when the bulk of the values for the vectors are zero.

### Distributed Representations of Sentences and Documents

Despite their popularity, bag-of-words representations have two major weaknesses: they lose the ordering of the words and they also ignore semantics of the words. For example, the words “cat,” “dog” and “Rome” are equally distant in a bag-of-word representation. This is analogous to the observations we made for the one-hot encoded representations of words. Indeed, as seen before, we can address these problems by resorting to distributed representations of documents. To this end, we can use algorithms that maps a variable-length document to a fixed-length distributed representation.

A simple strategy consists in representing the document as a concatenation or average of its surrounding words, and the resulting vector is used to predict other words in the context (Bengio, 2006). Then the embedded document representation can be exploited in downstream tasks, such as clustering and retrieval.

### Contextualized Document and Sentence Embeddings

In the previous sections, we have seen that we can derive contextualized representations of words using recent neural language models (Devlin, 2019).We can of course learn also contextualized representations of documents, i.e. fixed-length representations that derive from contextualized language models. For some time, transfer learning in NLP was limited to pre-trained word embeddings, but recent work has demonstrated strong transfer task performance using pre-trained sentence embeddings (Cer, 2018). The most commonly used approach is to average the BERT (Devlin, 2019) output layer or by using the output of the first token (the [CLS] token). However, this common practice yields rather bad sentence embeddings (Reimers, 2019). Other approaches have been investigated. One of the most used is Sentence BERT (Reimers, 2019), which is a modification of the pre-trained BERT network that use siamese and triplet network structures.

## Topic Models

In the following section each topic model will be described through mathematical notation, the graphical representation, that is Plate Notation and the generative process. This representation is widely used in probabilistic models to visualize repeated substructures within them, in this context is useful to understand frequentist models.

### Notation

Table 4.1 summarizes the most important mathematical notation that we use in this delivery.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

**Table 1**: Main notations for LDA and its extensions.

### Probabilistic Topic Models

Here, we will describe how the elements that compose a topic model can be interpreted in probabilistic terms, originating the prominent class of probabilistic topic models (Blei, 2012) (Zhai, 2017).

We have anticipated that topics are not just unsorted lists of keywords. Instead, they are associated with a weight or, rather, a probability weight. We can in fact express **a topic as a multinomial distribution over the vocabulary**, where the most likely words are the representative words of the given topic. We can therefore select the top-n most likely words to represent a topic.

In addition, topic models also provide a lower-dimensional representations of the documents in the space of the topics. Also, this representation can be interpreted as a probability distribution: **a document is in fact a multinomial distribution over the topics**. In other words, a document can talk about different topics in different percentages. Reporting the example mentioned in the Chapter 1, an NLP paper can talk about 30% of linguistic and 70% of computer science.

The only observations usually provided to a topic model are the documents and their words. We can imagine a generative process that have generated the words of the documents. A generative process is in fact the imaginary random process by which the model assumes the documents are constructed through the sampling of their words (observed random variables). The latent topics (latent random variables), which have ideally produced the collection of documents, are inferred by reversing the generative procedure of a text.

#### Latent Dirichlet Allocation (LDA)

To better explore these concepts, we now describe the most well-known topic model Latent Dirichlet Allocation (Blei D. M., 2003). It constitutes the base of probabilistic modeling, which assumes that observed data (documents) are generated from hidden variables (topics) following certain probabilistic distributions. In the context of LDA, each document is viewed as a mixture of topics (i.e. multinomial distributions over the topics), and each topic is characterized by a distribution over words (i.e. multinomial distributions over the vocabulary).

LDA is a **probabilistic graphical model**. Figure 3 reports LDA’s representation in plate notation. Here, the nodes of the graph represent the random variables and an edge between two nodes represents the conditional dependency relationships among the variables.



α

θd

znd

wnd

φ

β

Nd

D

K

**Figure 3**: LDA in plate notation. The variable wnd, representing the n-th word of document d, is observed, then it is represented by a gray circle. While the other variables are unobserved, thus they are represented by white circles. Variables are repeated if they are included in a rectangle.

Observed variables are represented by shaded circles (e.g. Wnd is the variable representing the words for LDA, which are observed, and in fact is represented by a shaded circle). Moreover, if a variable is contained into a plate, then the vari- ables are replicated multiple times (as the number reported on the corner of the plate).

More formally, let be K the fixed number of topics, D the documents and V the unique words of the vocabulary. In LDA, the only observations are the words w, and each word is associated with a topic assignment z. The topic assignments z are i.i.d (identically and independently distributed) drawn from a document-topic distribu- tion θ and word tokens are i.i.d. drawn from a topics’ distributions over words φ. In other words, the words of the documents in LDA are represented as BOWs, because the order does not count. The random variables θ and φ are multinomial distributions and are controlled by the Dirichlet priors α and β respectively.

The generative process of the documents in LDA is the following:

**for** each topic k *∈* K **do**

Draw a distribution over words ϕk|β ∼ Dir(β)

**end for**

**for** each document d *∈* D **do**

Draw a vector of topic proportions θd|α ∼ Dir(α)

**for** each word w in document d **do**

Draw a topic assignment znd|θd ∼ Mult(θd), where znd *∈* {1, . . . , K}

Draw a word wnd|znd, ϕznd ∼ Mult(ϕznd ), wnd *∈* {1, . . . , V}.

**end for**

**end for**

where Dir(*·*) and Mult(*·*) represent the Dirichlet and Multinomial distributions respectively. The full joint distribution of LDA, given its hyperparameters, is shown in the following equation:

p(θ, z, ϕ, w|α, β)

= p(ϕ |β)p(θ|α)p(z|θ)p(w|z, ϕ z)

document plate

= (znd|θd)p(wnd|znd,

word plate

Topic plate

topic plate

The goal is to compute the posterior distribution of the latent variables, given the observed documents. Therefore, the generative process of a document must be reversed in order to obtain the distribution of the hidden variables:

Immagine che contiene testo, Carattere, linea, bianco

Descrizione generata automaticamente

The denominator of this equation is intractable to be computed by means of exact inference methods. In fact, the posterior probability of LDA – and any other probabilistic topic model – is usually computed by approximate inference algorithms.

**TRAINING AND INFERENCE**

Different algorithms have been proposed to optimize inference for LDA. In the following, we will report the two main foundation methods that are used for training topic models. Although we will see the details of these methods applied for LDA, these methods can be generalized to most of probabilistic models.

* **Collapsed Gibbs Sampling (GS).** Markov Chain Monte Carlo (MCMC) methods are a class of approximate inference algorithms, which are scalable and allow sampling from a large class of distributions. One of the most widely used algorithms that belongs to this category is Gibbs Sampling (GS) (Bishop, 2006). The procedure starts from some initial state for the Markov chain, and, at each step, it replaces a value for i-th variable with the value drawn from the distribution of that variable conditioned on the values of the remaining variables. This procedure is repeated through all the variables, for T steps. Given p(z) = p(, , …, ) to sample, Gibbs sampling procedure is shown below:

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

After sampling enough samples, GS is proved to converge to the exact posteriors. However, it is hard to estimate how many iterations are required for the algorithm to converge. Choosing a number of iterations which is too high can lead to a computationally demanding execution for large-scale applications.

CGS can be employed for LDA (Griffiths, 2004), as both document-topic distribution and word-topic distribution can be calculated using just the topic assignments **z**.

Instead of the full joint distribution p (z, w, θ, ϕ|α, β), and are first integrated out, obtaining the distribution p(z, w|α, β) as it follows:

Immagine che contiene testo, Carattere, schermata, diagramma

Descrizione generata automaticamente

Since and only appear in the first and second terms, respectively, these integrals can be performed separately. The joint distribution marginalized over and then becomes:

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

The goal of Gibbs sampling is to approximate the distribution p (|w, α, β). The resulting collapsed Gibbs sampling equation for LDA can be written as

Immagine che contiene Carattere, testo, linea, tipografia

Descrizione generata automaticamente

where the superscript ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­¬nd signifies leaving the nth token of the d-th document out of the calculation. This notation will be used throughout this thesis. For simplicity, the hyperparameters α and β are assumed to be symmetric. Further details about collapsed Gibbs sampling for LDA can be found in (Griffiths, 2004).

* **Variational Inference.** Variational inference is a class of deterministic approximate techniques (Sun, 2013), which are an alternative to MCMC methods. The variational approach aims to approximate a posterior distribution by looking for a distribution from a family of distributions which is tractable and is the closest to the true posterior, where the closeness is generally measured by the relative entropy (or Kullback-Leibler divergence). More formally, the aim is to minimize the kl-divergence between the true posterior p(z|x, α) and the family of distributions p(z|v):



where **x** and **z** are arrays of observations and latent variables respectively, α is an array of fixed parameters and v is an array of free variational parameters. The log-likelihood of p can be limited by using Jensen’s inequality:

Immagine che contiene Carattere, testo, linea, diagramma

Descrizione generata automaticamente

Thus, Jensen’s inequality provides a lower bound, also called evidence lower bound (elbo), over the log-likelihood for an arbitrary variational distribution q(z|v):

It can be shown that maximizing the lower bound is equivalent to minimizing the kl-divergence. The optimal values for the parameters v can be obtained using standard nonlinear optimization techniques (Bishop, 2006).

LDA’s inference issue can be solved by using a mean field variational approach. By dropping the problematic edges and nodes, the simplified variational distribution is as it follows:

where γ and τ are the variational parameters. The next step is to find the optimal values for γ and τ, by solving the optimization problem explained above.

Variational inference is by far faster than MCMC methods. However, a variational method does not reach convergence, since the selected distribution is just an approximation of the true posterior, and finding a proper family of distributions can be difficult.

A relevant feature of probabilistic topic models is that they are modular and can therefore be extended.

### Hyperparameter Selection

An important element of topic models is the hyperparameters. First, it is important to clarify the distinction between parameters and hyperparameters, which are two recurrent terms in machine learning models. A *parameter* is a internal variable of the model that can be estimated or learned from the data. On the other hand, model’s *hyperparameters* cannot be learned during training but are set beforehand.

For example, let us consider LDA. The word-topic and the document-topic probability distributions are parameters of the model, because we estimate them during the training. While the number of topics is a hyperparameter because we have to set it before starting the training. Hyperparameters strongly affect the results and performance of a model. Let us imagine comparing the results of a topic model with 2 topics and a topic model with 500 topics, run on the same corpus. The first model will return coarse-grained and separated topics, while the other model will return finer grained and possibly overlapping topics. It is then important to carefully select the hyperparameters by adopting an appropriate search strategy, which is computationally tractable and effective.

Immagine che contiene Carattere, testo, bianco, tipografia

Descrizione generata automaticamenteLet us assume we must find the optimum of an unknown objective function f. Then, we are considering the problem of finding a global maximizer (or minimizer) of f:

where X is a design space of interest. This space can be composed of hyperparameters of different types: categorical, continuous or also conditional inputs.

In the following, we will talk about the most well-known techniques for selecting the hyperparameters in machine learning. We focus in particular on Bayesian Optimization, since it is a technique, we will extensively use in following Chapters.

#### Grid Search

The traditional way of performing hyperparameter selection is grid search. It consists in an exhaustive search of the hyperparameters through a manually specified subset of the hyperparameter space X. Grid search exhaustively considers all parameter combinations, Then the selected hyperparameter configuration is then one that returned the best results, according to a performance metric.

Grid search is reliable on low-dimensional space (1-dimensional or 2-dimensional) but suffers from the curse of dimensionality (Bergstra, 2012). However, this technique can be parallelizable because the hyperparameter settings it evaluates are typically independent of each other.

#### Random Search

Random Search instead selects the hyperparameter configurations to test randomly. This approach generalizes to continuous and mixed spaces. It can outperform grid search (Bergstra, 2012). Random search is in fact more efficient than grid search in highdimensional spaces if the objective function can be approximated by another function with less variables (hyperparameters). If the researcher could know ahead of time which subspaces would be important, then they could design an appropriate grid, however this is not always feasible.



**Figure 4**: Grid and random search of 9 trials for optimizing a function with

Figure 4 shows a comparison between a grid search approach and a random search approach, when two hyperparameters are involved and one of the two hyperparameters (unimportant parameter) does not change the objective function. It is evident that 6 of the total 9 trials of grid search are indeed ineffective.With random search, all the trials explore distinct values of the objective function. This failure of grid search is the rule rather than the exception in high dimensional hyperparameter optimization.

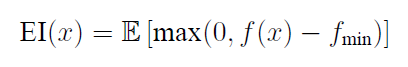
Random Search is also parallelizable, and additionally allows the inclusion of prior knowledge by specifying the distribution from which to sample.

#### Bayesian Optimization

Bayesian Optimization (BO) is a powerful model-based approach for optimizing expensive and noisy black-box functions, where the objective function is unknown and costly to evaluate (Archetti, 2019). The core idea of BO is to use all the configurations of the model evaluated so far to build a probabilistic surrogate model that approximates the objective function. This surrogate model, typically a Gaussian Process (GP), captures the uncertainty in the objective function based on observed data and is updated as new data points are evaluated.

**Figure 5**: Bayesian Optimization

The optimization process in BO is sequential and consists of selecting the next promising configuration to evaluate based on the current surrogate model. This selection is guided by an acquisition function, which balances exploration and exploitation. The acquisition function leverages both the mean and uncertainty (i.e., standard deviation) of the surrogate model to decide where to sample next. A common acquisition function used is Expected Improvement (EI), which aims to find points that are expected to improve upon the current best observation. Mathematically, EI is expressed as:



where ​ is the minimum observed value of the objective function so far, and E denotes the expectation.

The Bayesian Optimization process can be summarized in the following steps:

1. **Initialize the surrogate model**: start by building a probabilistic model, typically a Gaussian Process, using initial observations of the objective function.
2. **Acquisition function optimization**: based on the current surrogate model, optimize the acquisition function to select the next point to evaluate. The acquisition function balances the need for exploration (sampling areas with high uncertainty) and exploitation (sampling areas likely to contain the optimum).
3. **Evaluate the objective function**: evaluate the objective function at the selected point, obtaining a new observation.
4. **Update the surrogate model**: incorporate the new observation into the surrogate model, refining the posterior distribution of the objective function.
5. **Repeat**: continue this process until a termination criterion is met, such as a predefined number of iterations, a threshold on the uncertainty of the surrogate model, or reaching a desired level of accuracy.

The iterative nature of Bayesian Optimization allows it to efficiently navigate the search space of complex, costly functions by focusing evaluations on the most promising areas, leading to faster convergence compared to traditional optimization methods.

### Neural Topic Modeling

In recent years, neural topic models have gained increasing success and interest (Zhao, 2021), due to their flexibility and scalability. Several topic models use neural networks or neural variational inference (Ding, 2018).

Traditional approximate inference methods (e.g. mean-field and collapsed Gibbs) have the drawback that applying them to new topic models, even if there is a small change to the modeling assumptions, requires re-deriving the inference methods, which can be time consuming, and limits the ability of practitioners to freely explore the space of different modeling assumptions. AutoEncoding Variational Bayes (AEVB) (Kingma, 2014) seems to be a natural choice for topic models, because it trains a neural network that directly maps a document to an approximate posterior distribution, without the need to run further variational updates. Then, from this distribution, we can sample a lower-dimensional document representation. A decoder network (generative model) reconstructs the original input. We also call this architecture Variational AutoEncoder (VAE). In general, the input of these models is the BoW vector representation of the documents. We now focus of the main neural topic modeling approaches of the state of the art.

#### Neural Variational Document Model (NVDM)

The Neural Variational Document Model is composed of a Multi-Layer Perceptron (MLP) encoder (inference network) that compresses the input BoW document representation into a continuous latent distribution and a softmax decoder (generative model) reconstructs the document by generating the words indepedently.

Each word is generated directly from the dense continuous lower-dimensional document representation, sampled from the learned distribution. Figure 6 sketches the architecture of NVDM.

**Figure 6:** High-level schema of Neural Variational Document Model.

More formally, the authors define a generative model with a latent variable Let d ∈ be the bag-of-words representation of a document (where V is the size of the vocabulary) and ∈ be the onehot representation of the word at position i. An MLP encoder q(h|d) compresses document representations into continuous hidden vectors (d -> h). Then, a softmax decoder p(d|h) = reconstructs the documents by independently generating the words (h -> ).

τ parameterizes the generative distribution (d|h), while v are the inference network parameters.

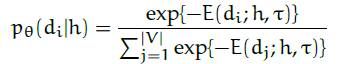
Immagine che contiene Carattere, testo, bianco, linea

Descrizione generata automaticamenteNVDM and LDA share the same generative process, with the exception that NVDM requires a Gaussian prior over the document-topic representation of the documents instead of a Dirichlet prior for computational reasons. The variational lower bound L is derived as:

where is the number of words in the document and p(h) is a Gaussian prior for h. During training, the model parameters τ together with the inference network parameters v are updated by stochastic back-propagation based on the samples h drawn from (h|d).

For the gradients with respect to v, the authors reparameterize h = μ + σ · ∈ and sample ∈ ~ N(0, I) (Kingma, 2014). And then the update of v can be carried out by back-propagating the gradients w.r.t. μ and σ. Based on the samples h ~ (h|d), the lower bound L can be optimised by back-propagating the stochastic gradients w.r.t. θ and ϕ. Since p(h) is a standard Gaussian prior, the Gaussian KL-Divergence can be computed analytically to further lower the variance of the gradients.

Immagine che contiene Carattere, testo, bianco, calligrafia

Descrizione generata automaticamenteThe conditional probability over words (i.e. the decoder network) is modeled by multinomial logistic regression:

where R ∈ represents the topics representations and represents the bias term. Let us notice that we can extract the topical words from the matrix R: the words of the vocabulary with the highest weights represents the most significant words of the topic k.

NVDM stands at the basis of different extensions of neural topic models. In the following we will focus on two of the most prominent NVDM extensions.

#### Embedded Topic Model (ETM)

The Embedded Topic Models (Dieng, 2020) aims to combine the benefits of LDA and word embeddings. It represents words and topics in the same embedding space. In particular, it embeds the vocabulary in an L-dimensional space (thus obtaining classical word embeddings). But also, a topic k is a vector k 2 RL. Therefore, k is a topic embedding, i.e. a distributed representation of the k-th topic in the semantic space of words. In its generative process, the ETM uses the topic embedding to form a topic distribution over the vocabulary. Specifically, ETM uses a loglinear model that takes the inner product of the word embedding matrix and the topic embedding. With this form, the ETM assigns high probability to a word v in topic k by measuring the agreement between the word embedding and the topic embedding. More formally, let p the Lx |V| word embedding matrix. In the generative process of ETM, a word is sampled according to softmax(), where is the topic assignment sampled from the document-topic distribution of document d.

Another difference between ETM and NVDM is that the NVDM uses a document real-valued latent vector, instead of a probability latent vector (as LDA). On the contrary, ETM constrains the latent variable h to lie in the simplex (its values are non-negative and sum up to 1).

In addition, ETM can automatically learn the word embedding representations or use pre-trained word embeddings. The use of pretrained word embeddings allows the model to add general information and improve the coherence of the topics over ETM with learned embeddings.

#### Product of Experts LDA (PRODLDA)

ProdLDA addresses two main issues in NVDM. The first challenge is related to the prior over the latent distribution of the document. NVDM uses a Gaussian distribution because it can be reparameterized, as we have seen before. To truly translate LDA into a neural topic model, we should assume a Dirichlet prior over the document. Yet the Dirichlet prior is not a location scale family, and that hinders reparameterization. To address this problem, ProdLDA explicitly approximates the Dirichlet prior using Gaussian distributions. In other words, the authors use an encoder network that approximates the Dirichlet prior p (θ|α) with a logistic-normal distribution (more precisely, this is softmax-normal distribution).

Another well-known problem of NVDM is the phenomenon of component collapse, in which the encoder network becomes stuck in a bad local optimum in which all topics are identical. To address this issue, the authors used the Adam optimizer, batch normalization and dropout units in the encoder network.

Moreover, LDA and ETM models the distribution p(w|θ, ϕ) as a mixture of multinomials. (Srivastava, 2017) note that this assumption leads to predictions that are never sharper than the components (the topics) that are being mixed. This can result in some topics appearing that are poor quality and do not correspond well with human judgment. To address this issue, the authors replace the word probabilities with a weighted product of experts (Hinton G. E., 2002) which is capable of making sharper predictions than any of the constituents’ experts by definition. More formally, the word-topic distribution ‑ of LDA becomes an unnormalized weight matrix and therefore the conditional distribution of is defined as

where σ is a sigmoid function. This modification allows the topic model to obtain a drastic improvement in topic coherence.

## Evaluating a Topic Model

## Hyperparameter Selection in Topic Models

Concerning the problem of setting hyperparameters for topic models, researchers have adopted different strategies. They usually select a priori fixed values according to some domain knowledge. For example, in the case of LDA, several approaches (Bao, 2009) fix the values of the hyperparameters and according to the work of Griffiths and Steyvers (2004). However, it has been shown that the same values do not apply to every dataset (Wallach, 2008). Moreover, in most cases, there is no prior knowledge of the distribution of the topics over the corpus and this makes the choice of the hyperparameter configuration difficult. Therefore, researchers usually select the best configuration of the hyperparameters using grid search techniques (Harrando, 2021) (Pavlinek, 2017). These approaches are easy to implement, parallelizable, and accurate in low dimensional spaces, but they suffer from the curse of dimensionality, as the number of the possible configurations grows exponentially with the number of hyperparameters (Bergstra, 2012).Another option is to adopt fixed-point methods for estimating the hyperparameters of a topic model (Asuncion, 2009). The inference algorithm alternates between sampling latent topics and inferring model hyperparameters. However, not every type of hyperparameter can be estimated with these methods. With the advent of neural topic modeling, other types of hyperparameters need to be considered. These are mainly related to the network architecture. Bayesian Optimization techniques (Archetti, 2019) can be superior to point estimates and grid search techniques, and it is designed for expensive objective functions. Yet, a thorough investigation of BO methods in topic modeling is still missing.

# Bibliography

Archetti, F. a. (2019). Bayesian Optimization and Data. *Springer International Publishing*.

Asuncion, A. W. (2009). On smoothing and inference for topic models. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 27–34.

Bao, S. X. (2009). Joint emotion-topic modeling for social affective text mining. *Ninth IEEE International Conference on Data Mining*, 699–704.

Bengio, Y. S.-S.-L. (2006). Neural Probabilistic Language Models. *Springer Berlin Heidelberg, Berlin, Heidelberg*, pages 137–186.

Bergstra, J. a. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 281–305.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics. *Springer-Verlag New York*.

Blei, D. M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 993–1022.

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

Caliskan, A. B. (2017). Semantics derived automatically from language corpora contain human-like biases. Science, 356(6334):183–186.

Cer, D. Y.-C. (2018). Universal sentence encoder for english. *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 169–174.

Devlin, J. C. (2019). BERT: pretraining of deep bidirectional transformers for language understanding. *Association for Computational Linguistics.*, pages 4171–4186.

Dieng, A. B. (2020). Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 439–453.

Ding, R. N. (2018). Coherence-Aware Neural Topic Modeling. *Proceedings of the 2018 Conference on Empirical*, pages 830–836.

Ferrone, L. a. (2017). Symbolic, distributed and distributional representations for natural language processing in the era of deep learning: a survey.

Grave, E. B. (2018). Learning word vectors for 157 languages. *In Proceedings of the International Conference on Language Resources and Evaluation.*

Griffiths, T. L. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, 5228–5235.

Harrando, I. L. (2021). Apples to apples: A systematic evaluation of topic models.

Hinton, G. E. (1989). *Parallel distributed processing: Explorations in the microstructure of cognition, chapter Distributed Representations* (Vol. vol.1).

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 1771–1800.

Khattak, F. K.-P. (2019). A survey of word embeddings for clinical text. *Journal of Biomedical Informatics*, X, 4:100057.

Kingma, D. P. (2014). Auto-encoding variational bayes. *2nd International Conference on Learning Representations*.

Lenci, A. (2008). Distributional semantics in linguistic and cognitive. *Italian journal of linguistics*, 20(1):1–31.

Liu, N. F. (2019). Linguistic knowledge and transferability of contextual representations. *Association for Computational Linguistics.*, pages 1073–1094.

Mikolov, T. S. (2013). Distributed representations of words and phrases and their compositionality. *In Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Pavlinek, M. a. (2017). Text classification method based on self-training and lda topic models. *Expert Systems with Application*.

Peters, M. E. (2018). In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. *Association for Computational Linguistics*, pages 2227–2237.

Radford, A. K. (2021). Learning transferable visual models from natural language supervision. *In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, volume 139 of Proceedings of Machine Learning Research*, pages 8748–8763.

Radford, A. W. (2019). Language models are unsupervised multitask learners. *OpenAI blog,*.

Reimers, N. a. (2019). Sentence-BERT: Sentence Embeddings Siamese BERT-Networks. *In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3980–3990.

Srivastava, A. a. (2017). Autoencoding variational inference for topic models. *5th International Conference on Learning Representations*.

Sun, S. (2013). Review of Deterministic Approximate Inference Techniques for Bayesian Machine Learning. *Neural Computing and Application*, 2039–2050.

Vaswani, A. S. (2017). Attention is all you need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017. pages.

Yang, Z. D. (2019). Generalized autoregressive pretraining for. *In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019,*, pages 5754–5764.

Zhai, C. (2017). Probabilistic Topic Models for Text Data Retrieval and Analysis. *In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1399–1401.

Zhao, H. P. (2021). Topic modelling meets deep neural networks: A survey. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4713–4720.