# 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

4.2.1. Latent Dirichlet Allocation (LDA) 9

## 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](#_Toc172841164)

[**Figure 2**: An example of word vector representation generated from text. 4](#_Toc172841165)

## 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,

Topic plate

word 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.

# Bibliography

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

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.

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.*

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

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

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.

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.

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.