# 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 3

2.3. Contextual Embeddings 5

3. Document Representations 5

## List of Figures

[**Figure 1.1**: 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](#_Toc172618958)

[**Figure 2.1**: An example of word vector representation generated from text. 4](#_Toc172618959)

## Topic Modeling

 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 1.1**: 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 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 similar to "*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 is based on the assumption 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 2.1**: An example of word vector representation generated from text.

There are different ways to generate these representations. One of the most famous model 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

# Bibliography

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

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,*.

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.