Università degli studi di Padova
Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea Magistrale in Informatica

MASTER THESIS

A DEEP LEARNING APPROACH FOR DISCURSIVE REPERTOIRES PREDICTION IN ONLINE TEXTS

Candidate:
Michele Bortone

Supervisor:
Prof. Giovanni Da San Martino

Academic year 2021/2022

Abstract

In recent years, a variety of deep learning techniques have been applied to natural language processing (NLP), the branch of artificial intelligence that focuses on the machine's ability to handle, understand and derive meaning from human languages. In most of cases, in order to create high performance models, a large amount of data and a lot of computational power is needed. Recent powerful hardware and new techniques for learning language from large datasets, such as transfer learning, allowed to increasingly apply natural language processing methods in different new tasks and domains. These new generation models are capable of creating effective word numerical representations that can be effective in tackling many NLP tasks.

Despite these improvements, it is still difficult to produce high-level representations of sentences in a text. A recent research program, started at the University of Padova and called dialogical science, proposes a novel methodology for analyze texts. Its aim is to examine a text focusing on *how* natural language has been used to communicate a concept and the future impacts and reactions it may generate. The study of a text using dialogical science is performed through the detection of 24 text units, called discursive repertoires, which provide a high-level analysis of text.

The main objective of this thesis is explore how dialogical science could be used provide meaningful sentence-level representations for NLP tasks.. In order to reach the goal, the thesis proposes a deep learning pipeline to predict discursive repertoires from online texts and use them as features for downstream NLP tasks. It is empirically shown the effectiveness of this approach on subjectivity classification, polarity classification and irony detection. The state of the art on irony detection is outperformed using only this novel representation.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Giovanni Da San Martino, for his support and guidance throughout the development of this work. I also would like to thank all the people from the University of Padova who collaborated and helped me during the months dedicated to the thesis.

A special thanks goes to my entire family, especially my sister, Valentina, who kept me motivated and confident.

Many thanks to all my colleagues for the time spent in the university and the goals achieved together.

Last, but not least, I would like to thank the "Dietro Le Quinte" team, although there will be more and more kilometers between us, we know that we can count on each other.

14-07-2022

Michele Bortone

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The idea of having machines capable of understanding human language has always fascinated scientists. The set of techniques that focus on the machine's ability to understand, interpret and generate human language is called natural language processing (NLP). After a first period in which NLP was dominated by rule-based approaches, the breakthrough came with the application of new machine learning techniques, a class of algorithms that allows to learn from experience. In modern days, the high production of textual data from the Internet has increased the effectiveness of machine learning approaches and NLP applications became increasingly widespread. For example, some of its applications are voice assistants, automated translators, and text classifiers.

Like other artificial intelligence fields, NLP has obtained a significant performance improvement with the use of neural networks and deep learning techniques. The standard approach for supervised machine learning tasks was to train a neural network directly on the training set for the downstream task. As opposed to the standard training step, a more recent approach, called transfer learning, has divided the training phase into two steps. In the first step, the model is trained on a large corpus, in order to learn effective word representations from text. Then, the pretrained model is fine-tuned on the smaller training set for solving the supervised learning task. Numerous works have shown that this technique is effective on downstream NLP tasks because the statistical properties learned from a large dataset allow to produce useful vectorial representations at word level, i.e. word embeddings. Most advanced representations also take context into account.

The text representation problem is one of the most important in the NLP field, because it has to deal with ambiguity of language and many other problems. Ambiguity is present at word-level, such as cases of polisemy, that is, multiple meanings for a single word. In that case, pretrained models seem to offer an effective solution to overcome this class of problems. However, ambiguity is present also at sentence-level and more in general, high-level representations are more difficult to be generated. Some sentences do not have clear interpretations, for humans as well. Examples of sentence-level ambiguities are syntactic and semantic ambiguity. Semantic ambiguity is related to the context and interpretation of a sentence. For instance, ironic sentences can be ambiguous because they present a figurative language, not easy to interpret for a machine. Despite large improvements in NLP downstream tasks due to the use of large pretrained models, these models seem to be not so effective in cases in which is important to detect properties at an higher-level than words. This is due to the fact that good sentence-level representations would need pretrained models trained on order of magnitude more data.

From a linguistic point of view, there are many ways to perform high-level text analysis and extract linguistic properties from a sentence. In most of cases it is performed a content analysis, focusing on the meaning of a text. A research program born at the University of Padova, called dialogical science, proposes a different approach, focusing on other features of the text, confining the meaning to a secondary role, and extracting clear and easy to interpret properties from text. The main goal of dialogical science is to analyze the method in which the content of a text is conveyed, detecting its capacity to produce new different impacts and interactions with the other speakers. For instance, if we analyze from this point of view the sentence "All politicians are evil!", we can predict with a certain degree of accuracy that probably this person is not open to a discussion about his opinion, so we can exclude future interactions regarding a change of his position. Instead, a soft approach such as "We need a better political class" indicates an inclination toward collaborative approaches regarding the topic of the text. Hence, in the dialogical science, the analysis of a text focuses not on the meaning, which is pretty similar in the two sentences, but on the way a concept is expressed, to predict and anticipate future and possible impacts as answer to the way a concept is conveyed.

To apply this approach, simple text units are detected during the analysis, called discursive repertoires. The detection of discursive repertoires is the main task in performing this kind of analysis. Discursive repertoires are 24 different classes in which a piece of text can be categorized. So, during the process, the text is partitioned into different spans with an assigned class. Although each class presents some technical features, not easy to understand and interpret without a deep knowledge of dialogical science, these classes and their names are representative of the way a concept is expressed. Some examples are the class "opinion" related to subjective declarations, or "Non answer" indicating a lack of interest or thoughts about a topic. This explicit set of categories makes it easier to interpret which properties are useful in solving a problem independently of the context and the topic. In fact, numerous projects focused on this methodology are born to help people in different areas, independently from the context. During COVID-19, dialogical science has been used to study reactions of a community about the evolution of the ongoing pandemic and estimate its level of cohesion to fight the emergency. Moreover, a similar approach is applied to provide psychological support to cancer patients, during the postoperative period. These are only two examples that show a variety of application of the dialogical science.

The flexibility and generalization ability of this methodology could help to overcome the problem of high-level representations of texts. An automated detection of discursive repertoires in texts that come from other NLP tasks can provide high-level properties useful to solve that downstream task. On the other hand, from the dialogical science researcher's point of view, a machine learning approach can automate a time-consuming task, allowing to analyze a large quantity of texts and finding new interesting applications of this theory to help people.

1.1 Goal and contributions

In this scenario, the main goal of this thesis is to investigate possible positive impacts of dialogical science when used in the context of NLP tasks. The contributions of this thesis to the pursuit of the main goal are the following:

- *Problem modeling:* This thesis proposes an effective deep learning approach to solve the discursive repertoires prediction task. The problem is divided into two subtasks and solved independently. The first one deals with the detection of spans delimiting each repertoire. This is a sequence tagging task. The second one is a multi-class classification task. Given a span of text, the goal is to identify the repertoire in it;

- *Model evaluation:* This thesis proposes a critical analysis of metrics useful to evaluate the performance of the described approach. The analysis focuses on the independent evaluation of the two tasks, and on a global evaluation of the entire pipeline. For all the evaluations, it is shown their drawback for the task;

- *Demonstration of effectiveness on downstream tasks:* This thesis covers an analysis of the use of discursive repertoires as features to improve performance in more common NLP tasks. To verify the effectiveness, it is proposed a model that takes in input different representations of discursive repertoires as alternative to the raw text, to solve a task. Three tasks are considered: subjectivity classification, polarity classification and irony detection.

In order to describe in details the contributions of this document, its structure is organized as follows. Chapter 2 provides the necessary background on dialogical science, machine learning and NLP. The focus is on the deep learning technique used in the main models. Chapter 3 covers the implementation and experiments about the model for discursive repertoires prediction, and a presentation of the Hyperion dataset used to train models. Chapter 4 presents the text segmentation task as a subroutine of the dialogical science method. Chapter 5 is focused on the evaluation of the entire pipeline for the detection of discursive repertoires. Chapter 6 applies the previous work to a series of downstream NLP tasks and compares the performance of this approach with those already published. To conclude, chapter 7 covers the discussion about results obtained in relation to the goal and the contributions, and then a brief presentation of the future work to expand the project.

Chapter 2

Background

This chapter aims to provide an overview of the technical knowledge necessary to understand the techniques and the experiments that will be described in the next chapters. The first section is a brief presentation of dialogical science, focusing on its main features and methodologies to analyze interactions between humans through natural language. The chapter then focuses on natural language processing and the application of machine learning techniques to process and extract meaningful features from written texts.

2.1 Dialogical science

Human language in all its forms, written and spoken, is a fundamental part of human interactions. Linguistics is defined as the scientific study of human language. It has many practical applications that allow us to study human social behavior. Like other academic disciplines, linguistics is divided into different subfields that focus on different layers that compose our language: some of them focus more on the internal structure of signs used in a language, such as morphology and syntax, while others focus on the meaning of these signs, such as semantics and content analysis.

During an interaction between two human beings, the use of natural language allows one to describe reality in a certain way and can generate different impacts for the other person. For example, the sentence "I think that the COVID-19 pandemic is not a serious problem" will generate future interactions about this opinion. Instead, the sentence "COVID-19 pandemic is fake!" does not expect a generation of future interactions with the aim of demonstrating falsity of this statement, because of the speaker attitude. Therefore, these interactions can be analyzed not only in terms of the meaning of words in a text or in a verbal communication, but also in terms of possible following states of a discourse, called *discursive configurations*, which can be generated starting from the previous interaction.

Dialogical science, a research program born in Padova, is a discipline that deals with the creation of tools and methodologies to analyze and study discursive configurations generated by human interactions through natural (or ordinary) languages. Hence, its object of investigation is natural language as a tool for generating new reality configurations, in the sense that reality is what we can describe through language, and its aim is to analyze and measure uncertainty on the configurations generated. All the rules and methodologies used by a researcher to analyze a text according to dialogical science are enclosed in the M.A.D.I.T methodology [18] Methodology of Textual Computer Data Analysis).

Research about dialogical science has detected 24 text units that can be used to create a configuration, called *discurisive repertories* and described in section 2.1.1. Therefore, the M.A.D.I.T methodology is a procedure of logical rules focused on the detection of these units in a text, taking as a starting point what has generated the text, that is the *question*. This element plays an important role in the discursive configuration and can influence which repertoires will be detected. This method allows the analysis and prediction of positions, thoughts and reactions that a person can assume in the near future about a particular topic and situation. The procedure is performed manually by the researchers, so it is a time-consuming task due to the strict rules for the identification of discursive repertoires. One of the objectives of this thesis is to automatically detect discursive repertoires through machine learning techniques to allow to use them for solving downstream natural language processing tasks.

2.1.1 Discursive repertoires

As stated before, the M.A.D.I.T methodology focuses on the identification of 24 text units that can be combined to change, generate or maintain discursive configurations. These text units are called *discursive repertoires*. From the text data that describe the discursive configuration, the main task is to partition the text into a sequence of discursive repertoires. The finite set of discursive repertoires forms and describes all the rules of natural language use adopted by the speaker, combining them with each other and defining the configuration. The discursive repertoires are grouped into three different categories related to the capacity to generate new discursive configurations:

- *Mantainance repertories:* set of rules of use of the language capable of preserving the current discursive configuration;
- *Generative repertoires:* set of rules of use of the natural language capable of generating new discursive configurations or reconfiguration of previously available configurations;
- *Hybrid repertoires:* set of rules of use of the language with mixed capacity. They can be used both for maintaining or generate new configurations.

These three categories and all the repertoires that belong to them are presented and described in the semiradial table, shown in figure 2.1.

Each discursive repertoire has features that make it different from the others:

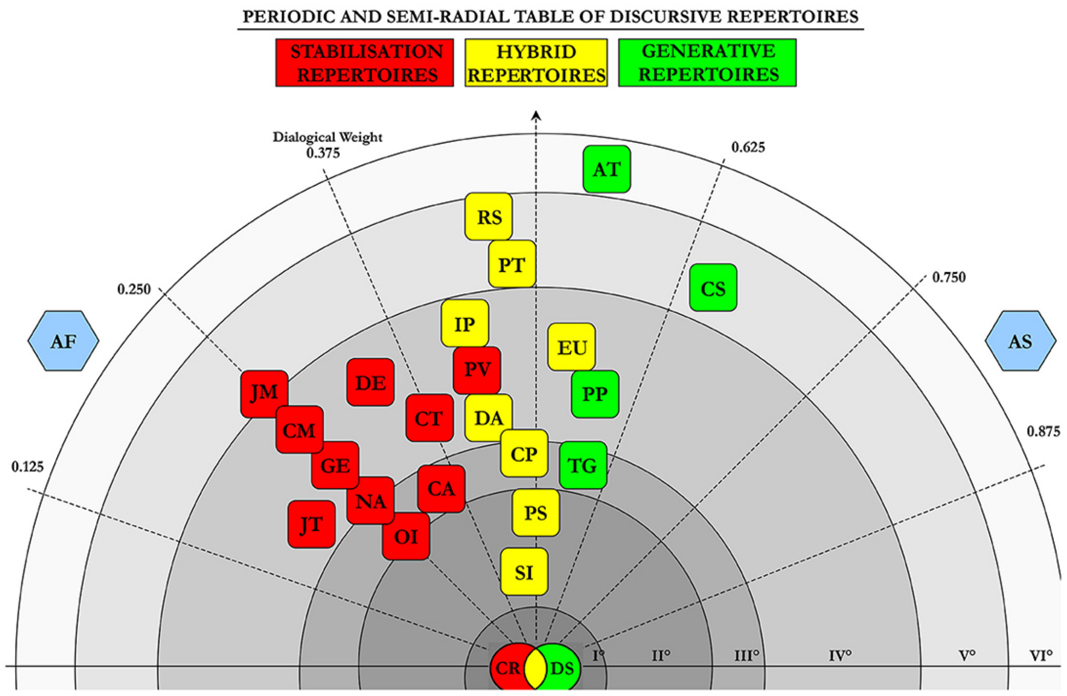


Figure 2.1: Table of discursive repertoires

- *Processual properties:* properties of a discursive repertoire. Each repertoire is made up of a different set of properties. A single property can belong to multiple repertoires, but a single set specifically belongs to a repertoire. All the processual properties are in relationship parent-child to some others, starting from primordial properties at the first level. Repertories at higher levels are composed of properties already available at the previous levels;

- *Dialogical weight:* It is a value assigned that means how much a repertoire is capable of constructing or generating a new discursive configurations. The higher the value, the higher its generation capacity.

In table 2.1, all the discursive repertoires and the correspondent category are shown.

There are some implications from the differences between repertoires shown in figure 2.1. Some repertoires are similar to others and quite different from another set. This is due to the category, the level and the dialogical weight. Hence, during a text analysis, the misclassification of a repertoire could be a minor or serious error depending on the mislead repertoires.

2.1.2 Practical applications

M.A.D.I.T methodology has been applied in different projects in the field of psychology and behavioral science with the aim of monitoring, analyzing the behavior of a community, and help members. Some examples are listed below:

Table 2.1: Discursive repertoires

Name	**Category**
Anticipation	Generative
Cause of action	Mantainance
Certify reality	Mantainance
Comment	Mantainance
Confirmation	Hybrid
Consideration	Generative
Contraposition	Mantainance
Description	Generative
Declaration of aim	Hybrid
Evaluation	Hybrid
Exemption from responsibility	Mantainance
Generalization	Mantainance
Judgment	Mantainance
Justification	Mantainance
Implication	Hybrid
Non answer	Mantainance
Opinion	Mantainance
Possibility	Hybrid
Prediction	Mantainance
Prescription	Hybrid
Proposal	Generative
Reshaping	Hybrid
Specification	Hybrid
Targeting	Generative

- *Hyperion observatory* [38]: During the COVID-19 pandemic, this project aimed to analyze through M.A.D.I.T methodology text written by citizens, journals, and institutions about the ongoing pandemic. Dialogical science is used to measure interactions between community members (citizens of Veneto, Italy) and to create an index that quantifies the cohesion of the community;

- *Persuasion index for fake news detection* [27]: The aim is to apply M.A.D.I.T methodology through machine learning techniques for fake news detection task;

- *Management of postoperative course in patients with cancer* [29]: This project aims to provide support during the postoperative period in cancer patients, improving the patient's competence in the management of postoperative daily life. The M.A.D.I.T methodology is applied by using questionnaires about quality of life;

2.2 Machine learning

Artificial intelligence is a large field with many practical applications and active research topics. This term refers to the computer science subfield that focuses on methodologies and algorithms capable of simulating characteristics of human intelligence, such as learning, perception and reasoning.

Machine learning is the artificial intelligence subfield that explores the ability of a human being to learn from experience. To solve a practical problem with machine learning, we need three main components:

- *Dataset:* a collection of *examples* that represents prior knowledge about the problem;
- *Model:* a statistical model able to recognize patterns on data and give predictions on new data unseen previously;
- *Learning algorithm:* a computational strategy to build the model exploiting the dataset. The core problem of machine learning is to train a model with a learning algorithm to solve a specific task.

A crucial element of the learning phase is *feedback*. Based on the feedback received during learning that is influenced by the different types of data available, we have four different learning strategies:

- *Supervised learning:* the dataset is a collection of labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$. in which each vector $\boldsymbol{x}_i$ is called *feature vector* and y_i is the label. A feature vector is a vector in which each dimension $j = 1, \ldots, D$ contains a value that describes the example somehow. Each variable can be either categorical or numerical. The label y_i can represent different types of data. It can belong to a finite set of classes $\{1, 2, 3 \ldots, C\}$, so in this case it refers to a categorical variable, or it can be simply a real number. A supervised learning algorithm exploits labels during training to produce a model capable of predicting labels taking new unseen feature vectors in input.
- *Unsupervised learning:* the dataset is a collection of unlabeled examples $\{\boldsymbol{x}_i\}_{i=1}^N$. In that case, we do not receive any feedback during the learning phase, due to unlabelled data.
  The goal of an unsupervised learning algorithm is to create a model that takes a feature vector $\boldsymbol{x}$ as input and either transforms it into another vector or a value that can be used to solve a practical problem. Examples of unsupervised learning tasks are *clustering* and *dimensionality reduction*.
- *Semi-supervised learning:* the dataset is mixed. It contains labeled and unlabeled data. A larger dataset represents better the probability distribution the data came from, so it is possible to help the model during the training phase adding unlabeled data to solve a supervised task.

- *Reinforcement learning:* in that case there is no dataset but an agent is trained exploiting its interactions with an environment. The idea is to learn a *policy*, a function that takes in input a feature vector and returns the optimal action the agent can do. To learn a policy, the agent receives a reward each time it performs an action and interacts with the environment. Examples of reinforcement learning are robot decision making and game-playing simulation, in which we have to compute sequential action predictions.

This work focuses on supervised learning applied on text, especially in classification problems.

2.2.1 Classification problems

A classification problem focuses on predicting the correct label of an unlabeled example. All tasks described in the following chapters are classification problems. Given a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ composed of labeled examples, the main goal is to generate a model capable of approximate as best as possible an unknown function f^*, which for a classifier $y = f^*(\boldsymbol{x})$ that maps a D-dimensional vector $\boldsymbol{x}$ to a label y from a finite set of possible labels. In most of cases the function f^* is not strictly deterministic, but represents a stochastic process, and the machine learning algorithm is used to learn the conditional probability distribution $P(y|\mathbf{x})$.

When we select a machine learning model, for example *Neural Networks* described in section 2.4 or *Support Vector Machines* reported in section 2.6, and then we train it, we select a function h that is the outcome of a search through a learning algorithm in a *hypothesis space* $\mathcal{H}$, the set of all possible functions the model selected can represent. In other terms, the goal is to define a mapping $y = f(\boldsymbol{x}; \theta)$ and choose the best set of parameters θ through a learning algorithm.

The simplest model implementable is a linear one:

$$f_{\boldsymbol{w},b}(\boldsymbol{x}) = \boldsymbol{w}\boldsymbol{x} + b \tag{2.1}$$

where $\boldsymbol{w}$ is a vector of weights and b is the bias term. Through a learning algorithm, the goal is to find the optimal set of *parameters* $\theta = (\boldsymbol{w}^*, b^*)$.

The function 2.1 returns output values in the range $[-\infty, +\infty]$. In order to perform a classification, we may need to normalize the output in a specific range, for example applying an *activation function.* In case of *binary classification* problems, a possibility is to use the *sigmoid* function (a.k.a. *logistic* function) that returns values in the range $(0, 1)$:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

The final model, known as *Logistic regression*, looks like this:

$$f_{\boldsymbol{w},b}(\boldsymbol{x}) \overset{\text{def}}{=} \frac{1}{1 + e^{-(\boldsymbol{w}\boldsymbol{x}+b)}} \tag{2.3}$$

Given a threshold t equal to 0.5 the class is positive if $f_{\boldsymbol{w},b}(\boldsymbol{x}) > t$ otherwise the class would be negative.

Generally, in case of *Multiclass classification* a model concludes with a *softmax* function:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.4}$$

where K is the number of classes. In the multiclass case, the output of the model will be a vector of probabilities that sums to 1.

This kind of algorithms can be considered *shallow*, in the sense that they can learn directly from the input features. The section 2.4 describes why neural networks allow the implementation of deep models, formed by many layers that take in input the output of preceding layers, and why this approach can be so powerful.

2.3 Natural language processing

Natural language processing is a branch of artificial intelligence and linguistics that focuses on the machine's ability to handle, understand and derive meaning from human languages in all their forms, both spoken and written. NLP tasks include low level ones, such as text tokenization and sentence boundary detection, as well as higher level ones, such as speech recognition, text summarization, name entity recognition, sentiment analysis and question answering

In recent years NLP has become crucial due to the enormous amount of text data produced every day through social networks, message apps, and in general all web platforms. These data cannot be analyzed by humans only, therefore we need to create pipelines able to automatically process text. The next sections will focus on how to process and represent text in a way that allows the machine to extract relevant information starting from raw data.

2.3.1 Text processing

Commonly, before tackling higher-level tasks, a set of low-level operations, which we refer to as text preprocessing pipeline, is first performed. The aim of a text preprocessing pipeline is to clean and highlight relevant information from raw and noisy data.

After a basic text cleaning, generally the first important step of a preprocessing pipeline is *tokenization*. The aim is to reduce unstructured text data to a sequence of meaningful elements, called tokens. Based on different tokenization techniques, a token could correspond to a character, a word as shown in figure 2.2, or more complex structures such as subword elements. The set of all tokens generated from the text forms a vocabulary. This phase is crucial not only for breaking unstructured text data but also for transforming them into fragments of information that can be considered as discrete elements. After obtaining a set of different symbols, it could be useful to reduce the number of possible tokens in the vocabulary. At this point, it is possible to apply basic

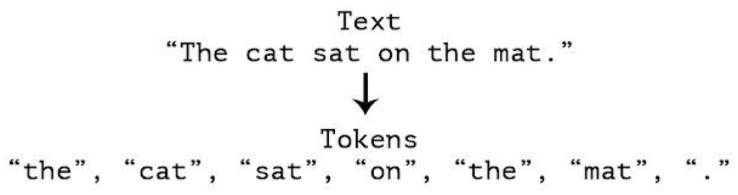

Text
“The cat sat on the mat.”
Tokens
“the”, “cat”, “sat”, “on”, “the”, “mat”, “.”

Figure 2.2: Example of tokenization at word level

techniques such as *lowercasing* all tokens, *removing stopwords* and *normalizing* symbols and numbers. Other possible approaches are to process the tokens sequence with more powerful techniques, such as *stemming* or *lemmatization*.

2.3.2 Text representation

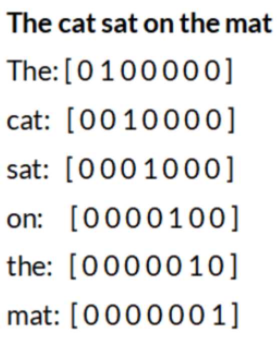
Raw text is composed of a set of symbols without a meaning for the machine. To build machine learning models capable of working with text data, we need first to create a representation of the text that contains its features. The techniques for creating a text representation can roughly be divided into two categories:

- *Sparse representations*: given a vocabulary, the text is represented as a vector with the same length as the vocabulary in which each element represents a feature related to a token. We can represent a text as a sequence of tokens using a *One hot enconding* as shown in figure 2.3, that is a vector set to 0 for all elements except the one representing the correspondent token, which is equal to 1. Alternatively, it is possible to represent the entire text as a single feature vector. Some examples are *Bag of words* and *TF/IDF*.

  Taking as reference the one-hot encoding, the main problem of this representation is that the vectors are sparse and each representation is equally dissimilar from the other, so this representation is unable to capture semantic similarities between tokens;

- *Dense representations*: they are a learned representation of all the tokens in the vocabulary through machine learning models such as neural networks. Generally, this kind of representation is referred to as *Word embeddings*. Word embeddings are dense fixed-size vectors of real values that can represent similarities between tokens. We expect words that are used in similar ways to result in having similar representations, naturally capturing their meaning. This is the key benefit in respect to feature-based representation such as Bag-of-words. Moreover, due to their dense representation, word embeddings have a low dimensionality that can help to solve common NLP tasks more easily. Famous neural-based representations are models *Continuous Bag of Words (CBOW)* and *Skip-gram* shown in figure 2.4.

  The main weakness of the two methods cited above is that they represent a word independently of context. We can think of a *context-independent* word embedding as a combination of all the senses that a specifc word can have, so this models cannot handle the word polisemy problem. Successive works explorate the possibility to solve this problem introducing *contextualized word embeddings.* Famous examples are ElMO [28] and BERT [12] models. Generally contextualized embeddings are produced with deep learning techniques that are covered in section 2.4.



The cat sat on the mat

The:	[0 1 0 0 0 0 0]
cat:	[0 0 1 0 0 0 0]
sat:	[0 0 0 1 0 0 0]
on:	[0 0 0 0 1 0 0]
the:	[0 0 0 0 0 1 0]
mat:	[0 0 0 0 0 0 1]

Figure 2.3: Example of One-hot encodings

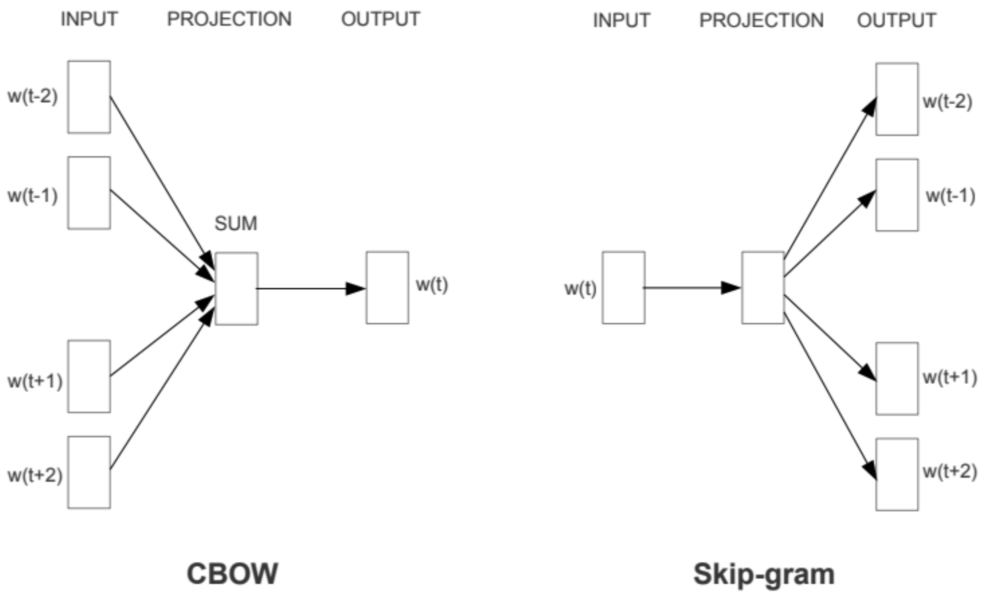


Figure 2.4: CBOW vs Skip-gram [24]

Currently, machine learning approaches are the state of the art in most of natural language processing tasks not only for text representation but also for classification and generation. Sections 2.4 and 2.5 cover the key points of recent techniques based on deep learning.

2.4 Neural networks and deep learning

In recent years, *neural networks* and *deep learning* have become hot topics, due to the ability to outperform many of the general machine learning models, especially when handling unstructured data such as text, audio and images.

A neural network is a mathematical function $y = f_{NN}(\boldsymbol{x})$ and in its shallow version for a classification problem is identical to the logistic regression model. However the typical neural network has a more complex structure, composed of nested functions:

$$f_{NN} = f_3(\boldsymbol{f_2}(\boldsymbol{f_1}(\boldsymbol{x}))) \tag{2.5}$$

Each of the nested vector functions are called *layers* and the last function f_3 is the *output layer.*
A single layer can be represented as this:

$$\boldsymbol{f}_l = \boldsymbol{g}_l(\boldsymbol{W}_l\boldsymbol{x} + \boldsymbol{b}_l) \tag{2.6}$$

Each function composing the vector is a linear combination between the parameters and the input that is passed to an *activation function* $\boldsymbol{g}_l$. Generally $\boldsymbol{g}_l$ is a non-linear function chosen based on the task. If we select a sigmoid, the model is identical to a logistic regression described in section 2.2.1. As stated before, generally neural networks are configured in a multi-layer structure, in which every layer takes as input the output of the previous layer. This architecture is called *feed-forward neural network* and can be represented organized in *layers* and *units* as shown in figure 2.5.

In 1987 it was shown that all continuous multivariate functions can be learned using a three-layer neural network [14]. In recent years, researchers explored different architectures composed of more than two hidden layers, and the term *Deep learning* refers to these approaches. The first question could be why prefer a deep network to a shallow one after results just described. Both shallow and deep model can approximate every continuous function but many empirical demonstrations have shown that a deeper model can achieve the same performance in terms of approximation accuracy of the shallower in a more efficient way. (i.e. using less units) [22].

In natural language processing, different deep architectures were proposed and outperformed other machine learning models. Deep neural networks reached the state of the art in almost any core NLP task, starting from language modeling, to word embeddings extraction, text classification and text generation.

2.4.1 Learning algorithms for neural networks

After building a neural network architecture, it is necessary to train the model using a learning algorithm. This section covers the key points about the training process. In section 2.2.1 it is claimed we want to find the set of parameters θ that best approximate a target function f^*. First of all, a *loss function* is needed to measure the performance of the neural network.

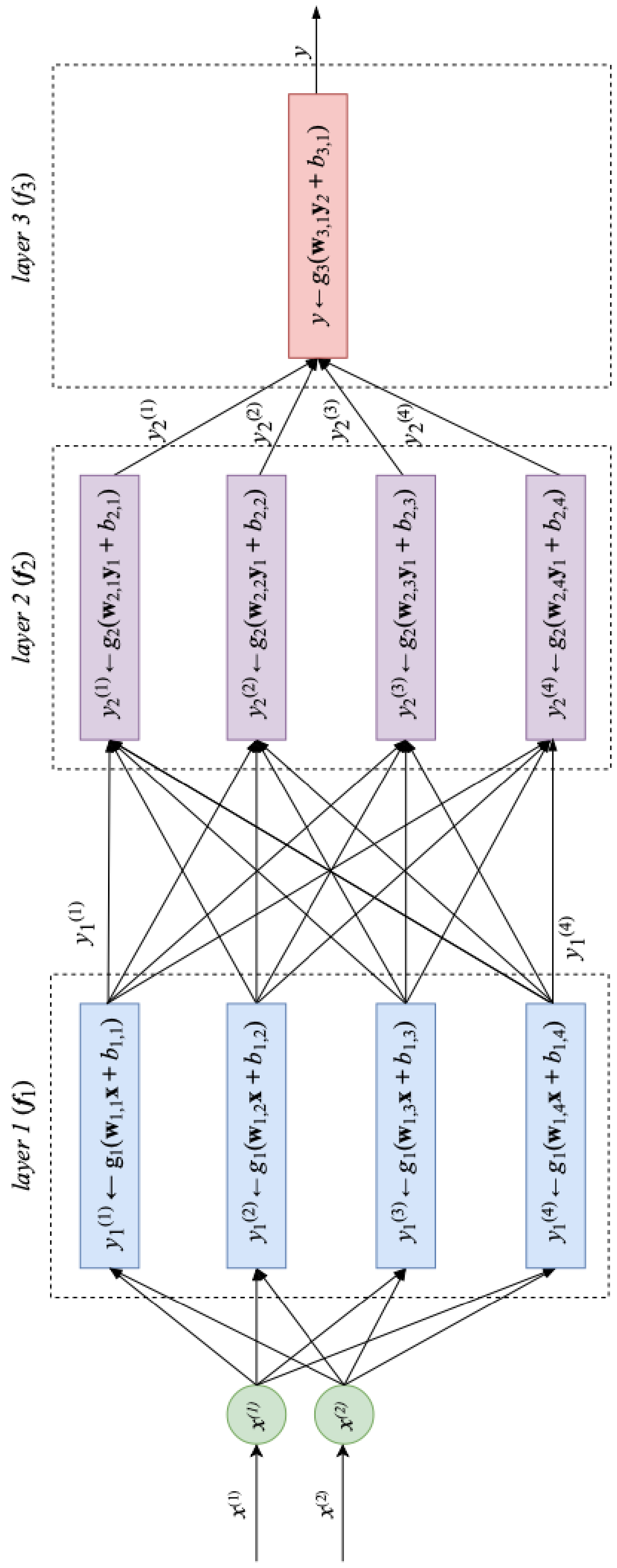

$x^{(1)}$
$x^{(2)}$
layer 1 (f_1)
$y_1^{(1)} \leftarrow g_1(\mathbf{w}_{1,1}\mathbf{x} + b_{1,1})$
$y_1^{(2)} \leftarrow g_1(\mathbf{w}_{1,2}\mathbf{x} + b_{1,2})$
$y_1^{(3)} \leftarrow g_1(\mathbf{w}_{1,3}\mathbf{x} + b_{1,3})$
$y_1^{(4)} \leftarrow g_1(\mathbf{w}_{1,4}\mathbf{x} + b_{1,4})$
$y_1^{(1)}$
$y_1^{(4)}$
layer 2 (f_2)
$y_2^{(1)} \leftarrow g_2(\mathbf{w}_{2,1}\mathbf{y}_1 + b_{2,1})$
$y_2^{(2)} \leftarrow g_2(\mathbf{w}_{2,2}\mathbf{y}_1 + b_{2,2})$
$y_2^{(3)} \leftarrow g_2(\mathbf{w}_{2,3}\mathbf{y}_1 + b_{2,3})$
$y_2^{(4)} \leftarrow g_2(\mathbf{w}_{2,4}\mathbf{y}_1 + b_{2,4})$
$y_2^{(1)}$
$y_2^{(2)}$
$y_2^{(3)}$
$y_2^{(4)}$
layer 3 (f_3)
$y \leftarrow g_3(\mathbf{w}_{3,1}\mathbf{y}_2 + b_{3,1})$
y

Figure 2.5: Example of multilayer neural network from [8]

An example of loss function for binary classification problems is *Binary Cross-*

Entropy:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i) \tag{2.7}$$

where y_i is the i-th prediction, $\hat{y}_i$ is the correct label and N is the the number of samples considered. Since the loss function computes the error rate of the

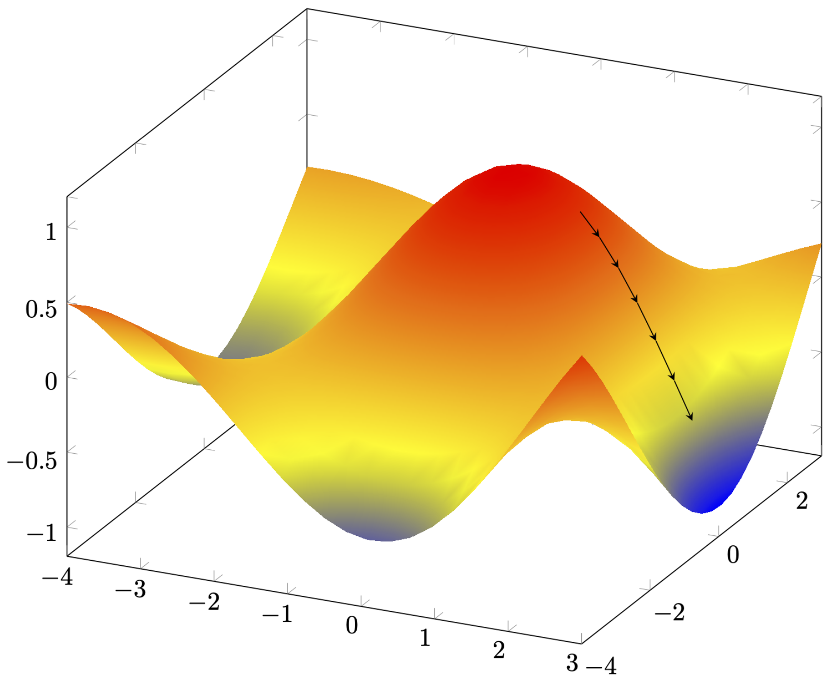


Figure 2.6: Gradient descent visualization

model, we select the parameters that minimize the loss function. To achieve this goal, the common strategy is *gradient descent*, an iterative optimization algorithm to find a *local minimum* of the function. Starting from a random weights initialization we compute gradients of the loss function to update the parameters proportionally to the negative of the gradient, doing a step towards the local minimum. This step is controlled by the *learning rate*, an hyperparameter that deals with the size of the update. A visualization of this idea is shown in figure 2.6.

Differently from evaluation metrics that are used to evaluate a model for a specific task after the training process, the loss function must be differentiable to compute gradients. Gradient-based learning can be applied to different machine learning models both linear and nonlinear, such as the case of neural networks. The main difference of using first-order methods such as gradient descent in neural networks is that nonlinearity causes loss functions to be probably non-convex, so there is no guarantee of convergence to the minimum. In addition, random initialization of weights can have a big impact on the training process. However, second-order methods for function optimization are not feasible in the case of neural networks, due to the large number of parameters.

In section 2.4 is stated that a neural network is a composition of nested functions, so the loss function becomes itself a nested function dependent on many parameters. A naive direct computation of all gradients would be very costly in computational terms, but in neural networks it is possible to apply *backpropagation algorithm* [36] for an efficient computation. After the *forward propagation* that allows to compute the output of a neural network, the backpropagation allows the information from the loss function to then flow backwards and propagating the error through the network, in order to compute the gradient. This operation is carried out efficiently using *chain rule of calculus* that allows to compute derivatives of compositions of functions.

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the inputs of each layer of a neural network, let $f_1 : \mathbb{R}^m \to \mathbb{R}^n$, $f_2 : \mathbb{R}^n \to \mathbb{R}$ be functions representing layers such that $\boldsymbol{y} = f_1(\boldsymbol{x})$, $z = f_2(\boldsymbol{y})$, so the chain rule application looks like this:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i} \tag{2.8}$$

Hence, if the neural network is composed of differentiable functions with known derivatives, it is possible to apply the chain rule and compute all the partial derivatives through the backpropagation algorithm and update the weights according to the loss function and perform gradient descent.

2.4.2 Recurrent neural networks

One of the most famous architectures used in NLP are *Recurrent neural networks* (RNN), a family of architectures capable of processing sequential data of variable length through recurrent layers. This feature is very important in common NLP task, either language modelling and sequence-to-sequence problems such as machine translation.

The main component of an RNN is the hidden unit $\boldsymbol{h}$. As shown in Figure

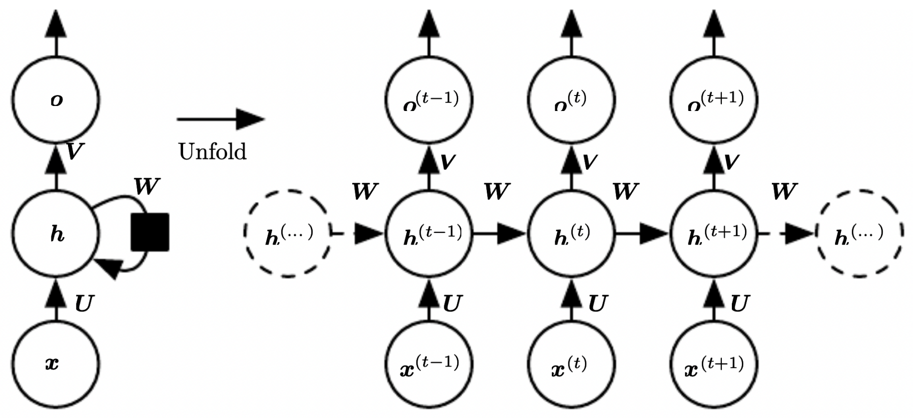


Figure 2.7: An unfolded view of the computationl graph of an RNN from [13]

2.7, $\boldsymbol{h}$ is defined as a function that combines at each step t the current input token x_t and the previous hidden state $\boldsymbol{h}_{t-1}$:

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}; \theta) \tag{2.9}$$

This formulation allows to combine contextual information to each token, maintaining always the same input size regardless of the sequence length. Another great advantage is the sharing of the set of parameters θ that belongs to f, in fact, the applied function is always the same at each step t. This implies a reduction of parameter with respect to a feed-forward neural network. Advanced implementations of RNNs such as LSTM-RNN [15] and GRU-RNN [9] have reached the state of the art in different NLP tasks.
This family of neural networks has remarkable advantages in handling sequences, but we have to manage some disadvantages due to their architectures:

- *Vanishing of gradients:* It occurs when the backpropagation algorithm moves back to compute gradients and update all parameters. Applying the chain rule to sequences with long dependencies, the gradients tend to "vanish" due to the multiplication of many gradients between 0 and 1. During backpropagation, the weights receive an update proportional to the gradient, so in that case the weight will not update significantly. An architectural strategy to solve this problem is to substitute activation functions with low derivatives such as the sigmoid function expressed in 2.2, using a different function with higher derivative such as ReLU (Rectifier Linear Unit):

$$f(x) = max(0, x) \tag{2.10}$$

- *Exploding gradients:* It refers to the opposite problem, always due to the multiplicative nature of the backpropagation algorithm. Gradients can become too large, so the weights update could skip a local minimum. This case is easier to handle, simply using a threshold and *clipping gradients* to an acceptable value;

- *Computational efficiency:* Working with long sequences and large vocabularies precludes parallelization and adds memory management problems. NLP tasks have many advantages from a very large training set and in that case learning from a large corpus can be very slow due to sequential computations.

Most of successive works on architectures for NLP tasks tried to solve the RNN problems described above.

2.5 BERT

In recent years RNN implementations such as LSTM and and GRU reached the state of the art in language modeling and sequence-to-sequence problems such as machine translation. In 2017 a research team proposed the *Transformer* [39], a not recurrent sequence-to-sequence model based on *attention mechanism*. Similarly to RNNs for language modelling the Transformer is composed of an enconder-decoder architecture as shown in figure 2.8. This new model allowed for a step up in terms of performance in different NLP tasks.

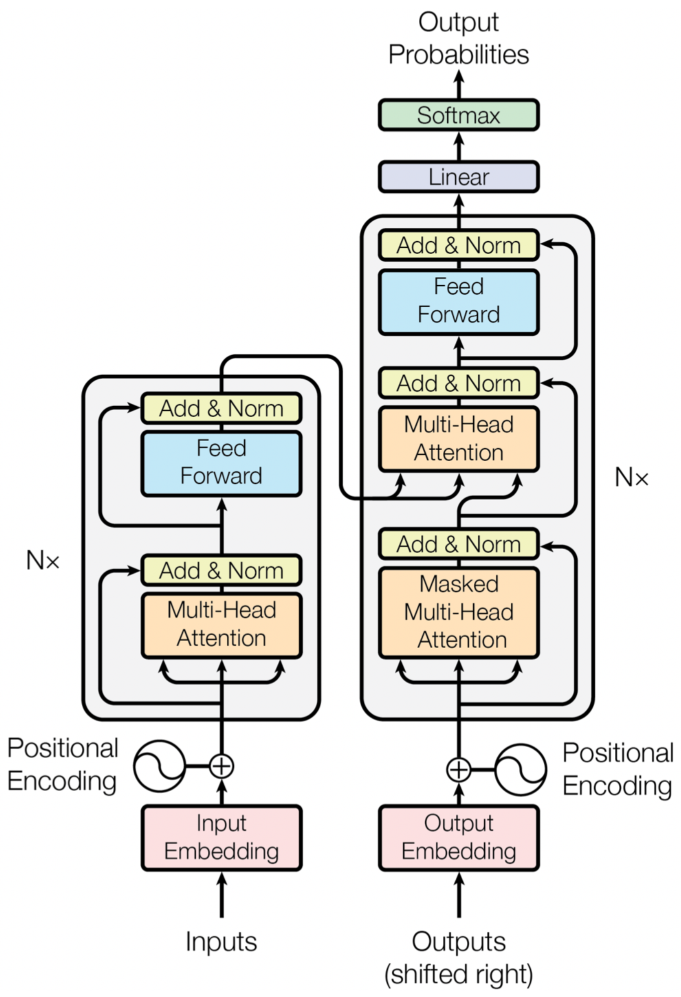


Figure 2.8: Encoder-Decoder architecture of the Transformer from [39]

During the following years, many research teams proposed their transformer-based architecture using the original model as a starting point. Table 2.2 shows the most important transformer-based architectures. In this work, BERT is widely used in the discursive repertoire prediction pipeline, both in text segmentation and in text classification. BERT stands for *Bidirectional Encoder Representations from Transformers* and presents some clever ideas regarding the learning process and architecture to produce contextualized embeddings and solve many NLP tasks.

2.5.1 Learning

The framework used by BERT to learn how to solve a specific task is called *transfer learning* and divides training into two steps. The idea is to train the model to solve different general tasks that enable to recognize patterns and features of the language. Then we can use this *pre-trained* model to initialize a new version of BERT that will be fine-tuned to make it capable of solving a *downstream task*, for example, text classification. Basically, the first step allows us to start from a good initialization of weights instead of using a

Table 2.2: Transformer architectures timeline

Name	**Year**	**Description**
Transformer [39]	2017	Encoder-Decoder model that leverages attention mechanism to produce better embeddings.
GPT [33]	2018	Stack of decoders from Transformer with unsupervised pre-training
BERT [12]	2018	Stack of encoders from Transformer with bidirectional attention
GPT-2 [34]	2019	Bigger and improved version of GPT
ROBERTA [19]	2019	Variant of BERT with different pre-training approach
GPT-3 [7]	2020	Bigger and improved version of GPT-2
Data2Vec [2]	2022	Multimodality Transformer-based architecture for language, speech and vision

random initialization.

Moving into the details, the pre-training step is made through *self-supervised learning*, a technique to train a model in a supervised task that does not need a human-labeled dataset. This technique allows to construct massive datasets to train the model because it does not need human help. BERT was specifically trained with the English Wikipedia dataset (about 2.5B words) and Google BooksCorpus (around 800M words). In the case of BERT, the pre-training process is composed of two tasks:

- *Masked language modeling:* The 15% of the input tokens are randomly masked with [MASK] special token and the model has to predict those hidden tokens;

- *Next sentence prediction:* It is a binary classifcation task in which a pair of sentences is provided to BERT. 50% pairs are made up of a sentence and the following. The other 50% the second sentence is randomly chosen from the corpus. This pre-training task allows to improve performances in tasks in which correlation between two sentences is important such as *question Answering.*

2.5.2 Model input

To allow the possibility of training BERT for solving different tasks, an input sequence for BERT can represent both one or two sentences, with a maximum length of 512 tokens. First of all, as described in Section 2.3.1 we need to tokenize the text in multiple tokens. The technique implemented in BERT is WordPiece [41], a tokenization algorithm that produces tokens at the subword level. The vocabulary has length l equal to around 30000 words with the addiction of some special tokens:

- *word*: It represents an entire word or punctuation or the first token of a word split in more than one token;
- *##subword*: It represents the following tokens of a word split in more than one token;
- *[CLS]*: Classification special token. It is added as the first token of the input sequence. In sequence classification tasks, its hidden state represents the aggregate result of classification;
- *[SEP]*: Separator token. It is added between the first and the second sentence;
- *[PAD]*: Padding token. Its utility is to fix the size of the sequence;
- *[MASK]*: Token used to mask other tokens in the Masked language modeling pre-training task;
- *[UNK]*: Token that substitutes a tokenized word not present in the vocabulary.

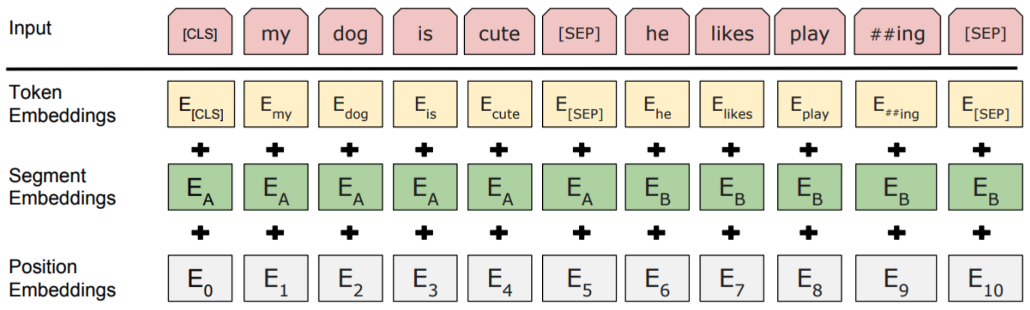

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.9: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. From [12]

After the tokenization process BERT need to receive in input a numerical form of the token. As shown in figure 2.9 the input representation for each token is the sum of three different vectors with dimension d:

- *Token embeddings:* They are fixed-size learned vectors that represent tokens. Similarly to Word2vec algorithms discussed in 2.3.2 this representation is not able to distinguish polisemy cases and the context, but in this case is only a starting point. The BERT embeddings layer with the aim of assigning the vector to a specific token can be represented as a lookup matrix with shape (l, d) randomly initialized and trained jointly to the model during the pre-training process;
- *Segment embeddings:* They are binary vectors to distinguish in which of the two input sentences the token is. Each vector contains zeros if the token belongs to the first sentence and ones otherwise;

- *Position embeddings:* Vectors representing the position of the tokens. Unlike RNNs that inherit the position information directly by processing recursively a sequence, BERT has no information about the position of a token. Transformer-based architectures need this information to be explicitly expressed in the input. In the original Transformer, position encodings are d-dimensional vectors generated through a fixed function. Instead, in BERT they are learned as well, so each of the 512 possible positions is a learned vector representation.

These three vectors are summed together and given as input to BERT.

2.5.3 Model architecture

Bert architecture is based on Encoder blocks of the original Transformer. Researchers proposed two different Bert models, a base and a large version.
Let L be the number of encoder blocks, H be the hidden states size and A be the number of self-atteention heads:

- BERT$_{BASE}$: $L = 12$, $H = 768$, $A = 12$, Total parameters $= 110M$;
- BERT$_{LARGE}$: $L = 24$, $H = 1024$, $A = 16$, Total parameters $= 340M$.

Due to limitations related to models pre-trained for the Italian language described in 2.5.7, this work is based on BERT$_{BASE}$. As shown in figure 2.10,

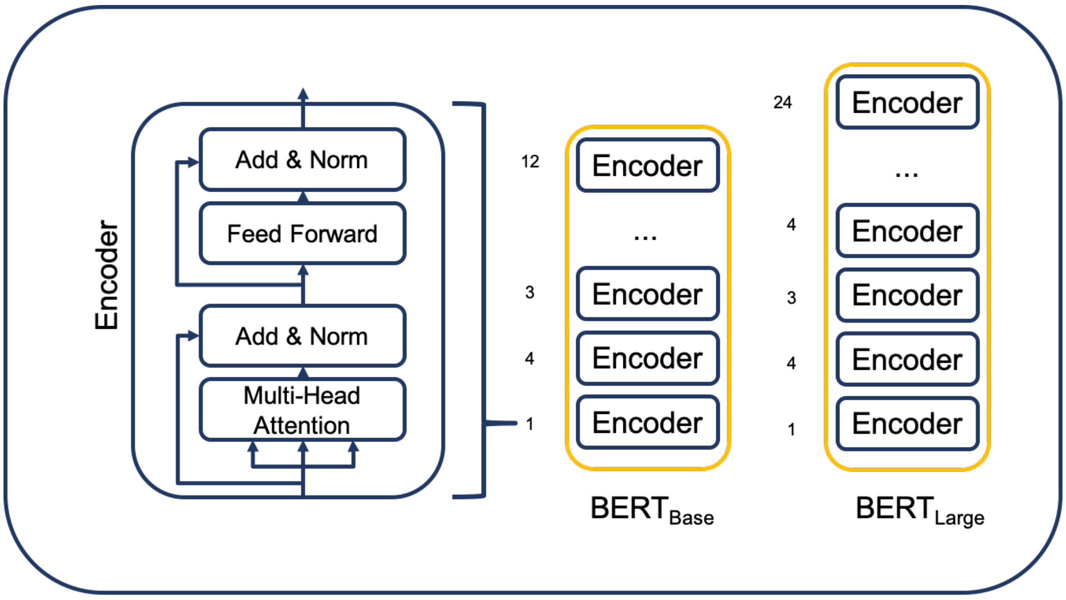


Figure 2.10: BERT encoder architecture

each encoder in BERT has the same structure as the encoder part of the Transformer. Starting from the first block, the sum of the three inputs flows and each encoder block is structured as follows:

- Multi-Head Attention: This is the most important layer of the model. It applies the attention mechanism to capture the context and relationships between tokens. It's called multi-head due to the fact that attention mechanism is applied multiple times in parallel. This particular process is described in Section 2.5.5;

- Residual connections: Each of the two sub-layers has a residual connection, that is a sum between the input and the output of a sublayer. Let $sublayer(x)$ be the output of each sublayer, so the input to the layer normalization will be $x + sublayer(x)$. Residual connections allow us to not lose input information in deep networks, such as the position encodings in the BERT case;

- Layer normalization [1]: It is a technique applied to normalize the distributions of intermediate layers. As opposite of batch normalization, it computes mean and variance to normalize across the features. The claim of authors is a better performance when dealing with sequences;

- Feed-forward layer: Normal fully connected layer to apply trainable weights and nonlinearity to the input.

2.5.4 Model output

The model output differs depending on the task in which BERT is applied. The standard version returns the hidden state vector in output from the last encoder, corresponding to each token in the input sequence.
These hidden states of the last layer of the BERT are then used for various NLP tasks. The common strategy is to add a new layer on top of BERT architecture to learn the specific task. In case of sequence classification tasks, the [CLS] token plays an important role, because its hidden state is the input of the classification layer.

In *next sentence prediction* the classifier layer performs a binary classification in the hidden state that corresponds to the [CLS] token, as shown in figure 2.11. Instead in *masked LM* a classifier with a softmax function returns a vector of probabilities with the length l of the vocabulary, to predict masked words.

2.5.5 Attention

Similarly to the cognitive process of attention, *attention mechanism* allows the network to focus on few relevant things and ignore things that are not relevant. In neural networks for NLP this technique allows to focus on tokens with some kind of relationship with others, capturing global dependencies inside the sequence. The first relevant model with attention was a recurrent neural network for machine translation [3], but this technique became a breakthrough with Transformer-based architectures.

Since BERT is composed of original Transformer encoders, it applies the same attention mechanism, called *self-attention*, because it is applied directly to the input, to choose which tokens to pay attention to. *Self-attention* is

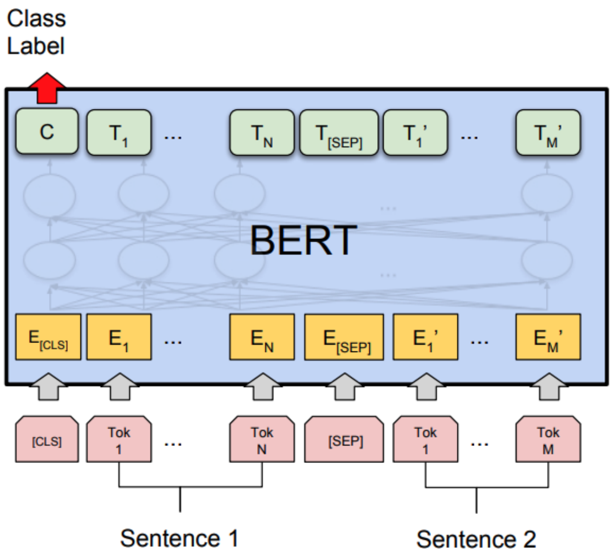


Figure 2.11: Next sentence prediction as a binary classification task with BERT model

defined as a function of *query, key, value* multiple vectors packed in three matrices. Let Q, K, V be the three matrices and d_k the dimensionality of k, the function is defined as follows:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.11}$$

In self-attention, all three matrices Q, K, V correspond to the input. The matrices Q and K are multiplied, then scaled to the value $\frac{1}{\sqrt{d_k}}$ and at the end multiplied with the values to obtain the output.
Figure 2.12 shows a visualization of the operations applied. The mask operation is not used in Transformer encoder, so neither in BERT. Before applying attention, Q, K, V (i.e. the input) are linearly combined with learned weights matrices W^Q, W^K, W^V, creating three different representations of the input for each head. BERT implements a *multi-head attention*, in the sense that these operations are applied multiple times in parallel. After that, the output of each head is concatenated and multiplied by a learned weights matrix W^O.

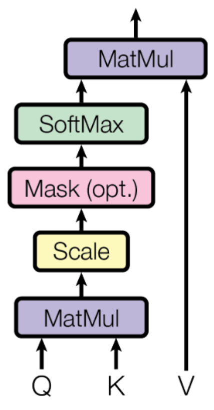
The entire process is defined as the following function:

$$MultiHead(Q,K,V) = Concat(head_1, \ldots, head_h)W^O \tag{2.12}$$

where $head_i$ is:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.13}$$

Although the attention mechanism seems complex, for each token, the output is a weighted sum of all the tokens. The weight assigned to each token is the output of the softmax function. As shown in figure 2.13, attention forces the model to focus on relationships between tokens.



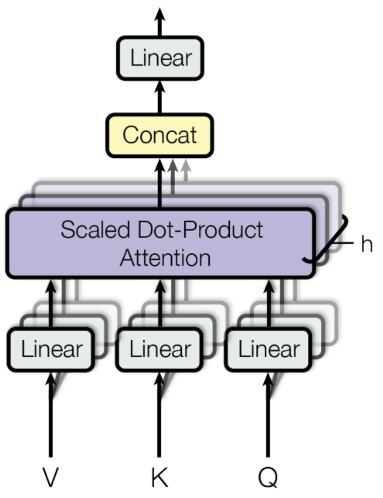


Figure 2.12: At the left, the specific self-attention mechanism applied by Transformer encoder architectures such as BERT, called Scaled Dot-Product Attention. On the right, a virtualization of the multi-head attention, a concatenation of multiple attention blocks

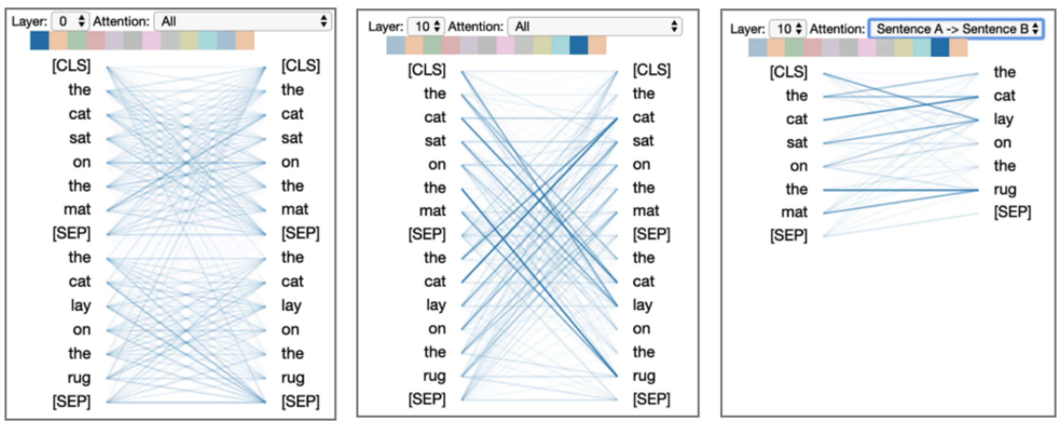


Figure 2.13: BERT self-attention heads visualization with BertViz [40]. The darker the connection the higher the softmax output

2.5.6 Contextualized embeddings extraction

The common approach to the BERT model is to fine-tune the model for a specific task. In that way, the model is able to produce contextualized embeddings that are the input for an added classification layer on top of BERT. This layer is randomly initialized and then is trained jointly with all the other layers during the fine-tuning process. In this section, another possible approach is discussed, called *feature-based approach.* As discussed, the BERT base model uses 12 layers of Transformer encoders, and each output per token from each layer of these can be used as contextualized embeddings.

One of the main advantages is the possibility of pre-computing the embeddings, so it is possible to use a more complex classifier instead of a single layer and not be forced to maintain the specific BERT architecture. The original authors proposed different methods to combine the encoders output, as outlined in Figure 2.14. The Chapter 6 describes the best techniques in terms of

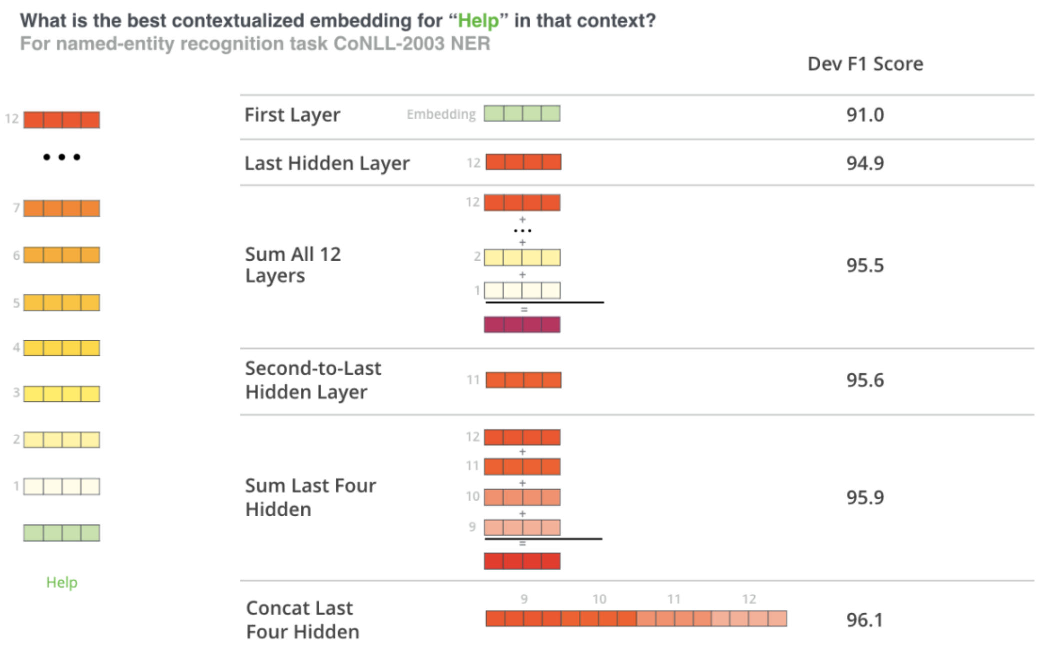


Figure 2.14: Techniques for contextualized embeddings extraction and results for *name entity recognition* task COLL-2003 NER

performance used in this work.

2.5.7 BERT for the Italian language

The original BERT model reached the state-of-the-art for many English tasks when it was released. The same authors released a multilingual version of BERT (M-BERT) trained on a large Wikipedia corpora in 104 different languages. Numerous studies demonstrate that M-BERT can perform quite well on cross-lingual tasks [30], and, more generally, this model can be a good baseline for tasks in a specific language different from English. Although these results, the original documentation of M-BERT showed that the performance in English tasks of M-BERT is worse in respect of the single language BERT, but can be very difficult to train and maintain a model for each language.

Successively, a lot of different teams around the world trained their single language BERT for many different languages. This happened also in Italy, mainly with two models used in this work:

- AlBERTo [31]: BERT version trained from scratch on TWITA [5], a collection of italian messages written on Twitter social network. This model is specifically trained with social network messages, and is able to handle typical symbols such as hashtags and emojis;

- DBMDZ BERT[37]: BERT version trained on OPUS corpora collection

(Wikipedia dump) and the Italian part of OSCAR corpus. This is a more generic Italian language model.

All these two model were tested during the experiments.

2.6 Support vector machines

Support vector machine is a linear model for both classification and regression. In a classification problem, given a dataset with dimension d, the idea behind SVM is to see data as a point on a d-dimensional space and draw a *hyperplane*, that is a $d-1$-dimensional object that separates the space in two parts, one with positive samples and one with negatives. The following equation represent the hyperplane:

$$\boldsymbol{w}\boldsymbol{x} - b = 0 \tag{2.14}$$

and a support vector machine for a classification task is defined as follows:

$$f(\boldsymbol{x}) = sign(\boldsymbol{w}^*\boldsymbol{x} - b^*) \tag{2.15}$$

where $\boldsymbol{w}^*$ and b^* are the optimal parameters and the *sign* function returns the predicted label (1 for positive, -1 for negative) and it is defined as follows:

$$sign(x) = \begin{cases} 1 & \text{if} \quad x >= 0 \\ -1 & \text{if} \quad < 0 \end{cases} \tag{2.16}$$

The core problem of this algorithm is to compute the optimal parameters $\boldsymbol{w}^*$ and $\boldsymbol{b}^*$ to find the hyperplane that correctly separates the samples in the dataset, so $\boldsymbol{w}$ and b must be chosen with respect to the following constraints:

$$\begin{aligned} \boldsymbol{w}\boldsymbol{x}_i - b \geq +1 \quad \text{if} \quad y_i = +1 \\ \boldsymbol{w}\boldsymbol{x}_i - b \leq -1 \quad \text{if} \quad y_i = -1 \end{aligned} \tag{2.17}$$

To improve generalization performance, we do not look for any hyperplane respecting this constraint, but we choose the one that maximizes *margin* from the classes, that is, the distance between the hyperplane and the closest example for each class.

As shown in figure 2.15 the distance between the two hyperplanes $\boldsymbol{w}\boldsymbol{x}_i - b = +1$ and $\boldsymbol{w}\boldsymbol{x}_i - b = -1$ is given by $\frac{2}{\|\boldsymbol{w}\|}$ where $\|\boldsymbol{w}\|$ is the Euclidean norm of $\boldsymbol{w}$, so to maximize the margin we need to minimize $\|\boldsymbol{w}\|$. For computational reasons, it is preferable minimize $\frac{1}{2}\|\boldsymbol{w}\|^2$.

Hence the quadratic optimization problem to solve looks like this:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & y_i(\boldsymbol{w}\boldsymbol{x}_i - b) \geq 1 \; \forall \{\boldsymbol{x}_i, y_i\} \in D \end{aligned} \tag{2.18}$$

This formulation is called *hard-margin SVM*, because the idea is to look for a hyperplane that perfectly separates the two classes. However, in most cases, this is not possible because the data are not linearly separable because of noise

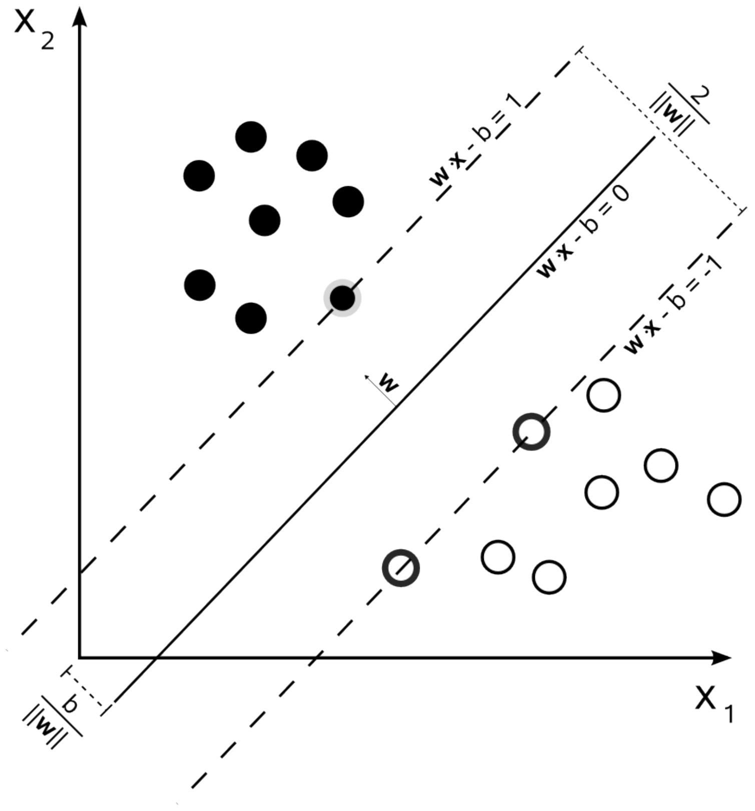


Figure 2.15: Support vector machine example. The closest samples to the hyperplane are called *support vectors*.

or because a nonlinear model is needed. A solution to increase model capability is to add the *Hinge loss* function as a term of the function to optimize. The Hinge loss in the context of SVMs is defined as follow:

$$hinge(\boldsymbol{x}_i, y_i) = max(0, y_i(\boldsymbol{w}\boldsymbol{x}_i - b)) \tag{2.19}$$

So, the *soft-margin* support vector machine looks like this:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & C\|\boldsymbol{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} max(0, y_i(\boldsymbol{w}\boldsymbol{x}_i - b)) \\ \text{s.t.} \quad & y_i(\boldsymbol{w}\boldsymbol{x}_i - b) \geq 1 \ \forall\{\boldsymbol{x}_i, y_i\} \in D \end{aligned} \tag{2.20}$$

where C is an hyperparameter that specifies the tradeoff between maximizing the margin and minimizing mistakes in classification of training data.

To manage with non-linearly separable data it is possible to use the *kernel trick*. The idea is to transform data into higher-dimensional samples to make them linearly separable, as shown in figure 2.16, using a function $\phi : x \rightarrow \phi(x)$.

In order to train a SVM capable of separating data in a higher-dimensional space, we need to compute many scalar products between features in the transformed feature space of ϕ. These operations lead the problem to impractical computational costs. To overcome this limitation, the function ϕ is not directly used, but it is defined a *kernel function* capable of computing the dot product

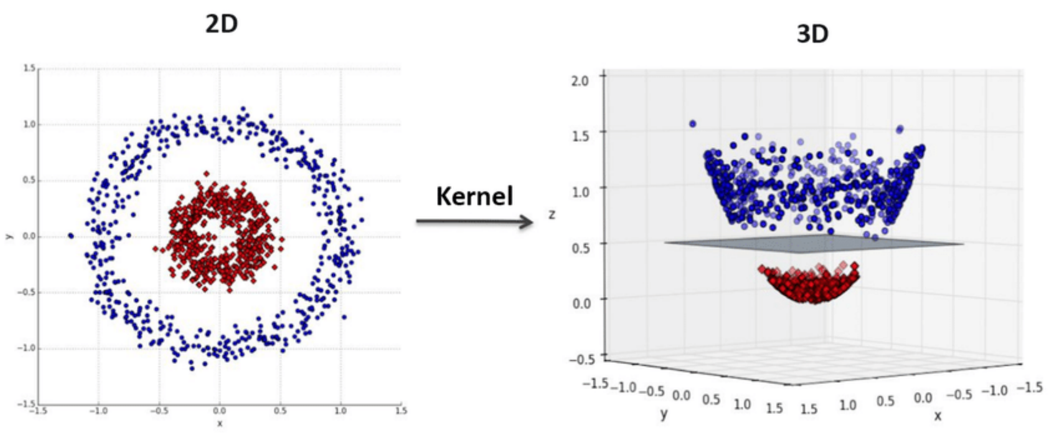


Figure 2.16: Kernel trick applied on bidimensional data [35]

between two higher-dimensionality vectors:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})\phi(\boldsymbol{x}') \tag{2.21}$$

In this way, it is not needed to manually compute the dot product, so the computation will be more efficient.

An example of a kernel function is *radial basis function* (RBF):

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma^2}) \tag{2.22}$$

where σ is an hyperparameter.
An alternative can be the polynomial kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T\boldsymbol{x}' + c)^d \tag{2.23}$$

Chapter 3

Discursive repertoires prediction

This chapter describes the deep learning approach used to predict a discursive repertoire starting from a text span. Before the task definition, the dataset used for the experiments (Hyperion) is introduced. Then, after the formalization of the task covered in section 3.2, section 3.3 and 3.3 focus on the machine learning techniques to solve the problem and on a critical discussion on the most effective metrics to evaluate it. The last part is related to a presentation of the experiments performed and the analysis of results.

3.1 Hyperion dataset

All the experiments described in chapters 3 and 4 are based on the *Hyperion dataset*, a human-annotated dataset created for the project "Hyperion observatory" described in section 2.1.2. In that project, 15,332 texts written by citizens, journals, and institutions about the ongoing COVID-19 pandemic, were analyzed to create a social cohesion index for the emergency management

As stated in section 2.1, the outcome of a text analysis through the M.A.D.I.T methodology is influenced by the *question*, so the distribution of discursive repertoires is strictly related to this element. In the Hyperion dataset the analyst classified the texts using only one question, the following:
What is the purpose of this text for the pursuit of the common goal of reducing the spread of the infection?
Hence, the analyzed text indicates how a person is willing to maintain a behavior useful to fight the pandemic.
Due to the context of project that has generated data, all the 15,332 texts are in Italian language and referred to a common topic, that is COVID-19 and related political strategies. Some of these key topics are shown in figure 3.1.

The texts belong to two different sources:

- *Twitter messages:* This type of text is short and presents special characters and words such as emojis and abbreviations;
- *Newspaper articles:* Long texts with correct and standard Italian grammar.

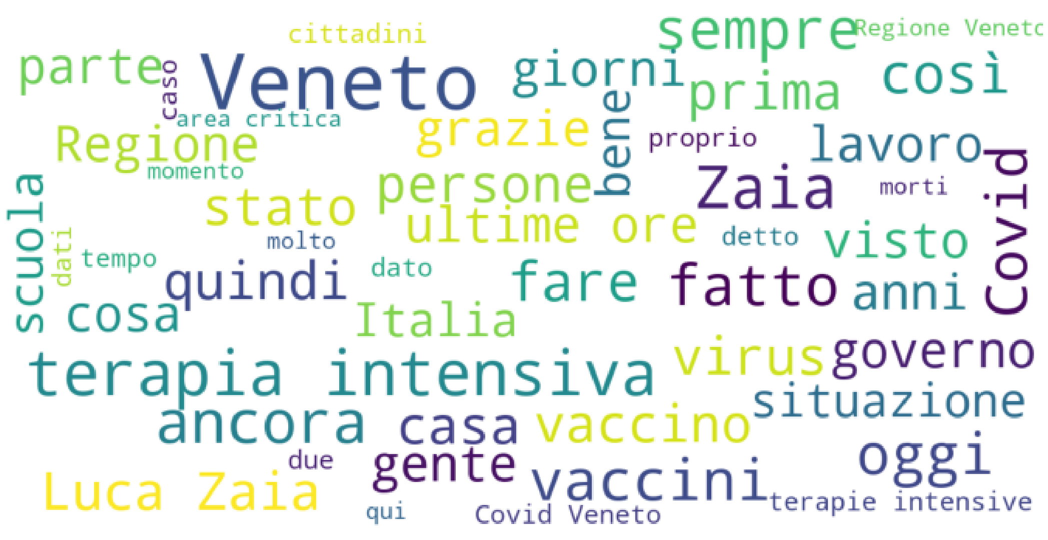


Figure 3.1: Word cloud of Hyperion dataset

Looking for *hashtags* # and short texts in the dataset it's been possible to estimate that tweets are about 95% of dataset, although the actual source was lost during data collection.

Texts in the dataset are composed as shown in table 3.1. Each text is partitioned into a sequence of spans, and then a discursive repertoire is assigned to each span. So when researchers apply the M.A.D.I.T methodology to a text, they extract a sequence of discursive repertoires. An exploratory analysis of the Hyperion dataset, described in table 3.2, allowed to show some features to take into account during the experiments. Regarding the number of spans, it follows that each text has good probabilities of not being divided into multiple spans, each referring to a different repertoire. This is due to the nature of the dataset mainly composed of short Twitter messages. The experiments made in chapter 4 take into account this particular feature.

The following analysis showed that it is necessary to deal with two main problems when experimenting with this dataset:

- *Noise:* It is expected that the set of span forms a partition of the text. Unfortunately, this is not the case, so to in a significant number of texts. To address this problem, the dataset is cleaned by reconstructing some texts as a concatenation of all spans that belong to the text.
  Moreover, there are frequent cases of wrong and double punctuation and cases of grammatical or general errors, so it is fairly common to produce tokens from the text that do not belong to the vocabulary. In this case also, the preprocessing phase has deleted some noise;

- *Unbalanced repertories:* As shown in figure 3.2, the dataset is pretty unbalanced. *Anticipation* repertoire is the least frequent with only 53 samples. Instead, the most frequent is *certify reality* with 5,627 samples. Furthermore, a repertoire is not present. There are some techniques that can be used to deal with this problem, such as oversampling and

Table 3.1: Hyperion dataset

Text	**Span**	**Repertoire**
Dunque vediamo se ho capito: Conte le canta a Salvini e alla Meloni, questi reagiscono e le cantano a loro volta a Conte, Mentana si smarca da Conte e finisce con foto celebrativa nelle pagine social della Bestia. Siamo tornati per un attimo alla normalità: meraviglia assoluta!	Dunque vediamo se ho capito:	Declaration of aim
	Conte le canta a Salvini e alla Meloni	Certify reality
	questi reagiscono e le cantano a loro volta a Conte	Justification
	Mentana si smarca da Conte e finisce con foto celebrativa nelle pagine social della Bestia.	Certify reality
	Siamo tornati per un attimo alla normalità:	Comment
	meraviglia assoluta!	Judgement

Table 3.2: Hyperion dataset statistics

Feature	**Result**
Number of texts	15332
Number of spans	35148
Number of repertoires	23
Spans average for each text	2.3
Percentage of single span texts	63%
Words average in a span	34

undersampling. In this work, a different approach has been used by customising the loss function used in the training phase. This solution is described in the following sections.

3.2 Task definition

When a researcher applies the M.A.D.I.T methodology starts from an entire text and looks for a sequence of repertoires. To do this, the text is divided into spans. The span can be defined as the text piece in which the repertoire does not change, so the text segmentation is performed at the same time as the repertoires detection. In this work, to replicate the researcher's analysis,

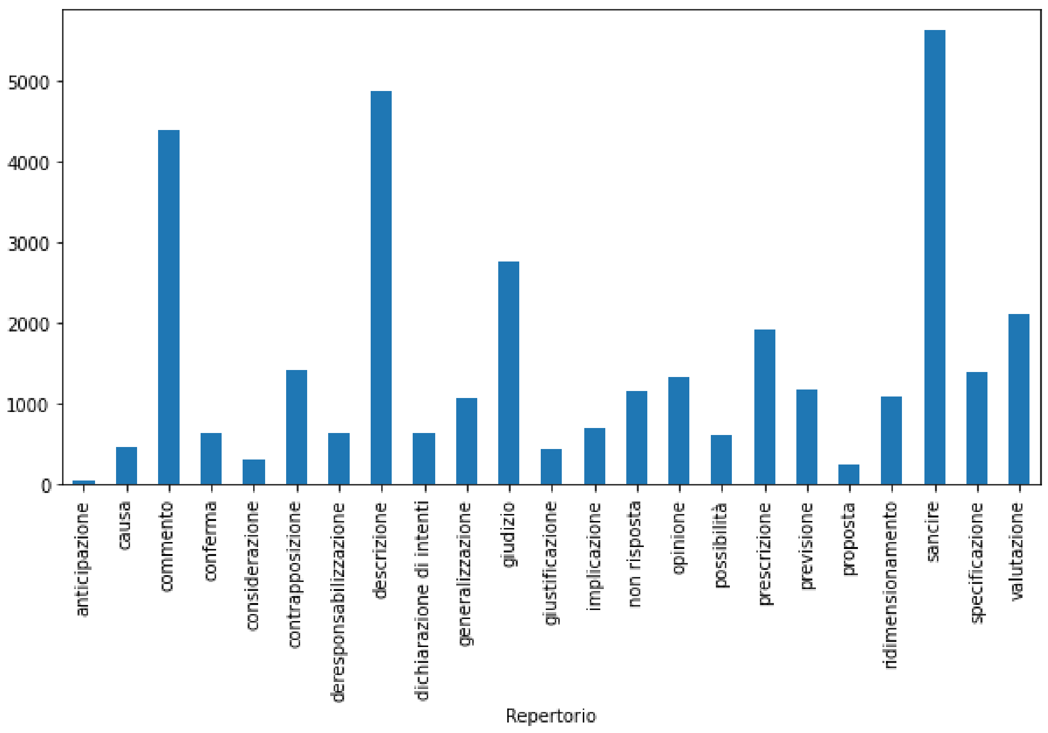


Figure 3.2: Discursive repertoires distribution

the problem is modelled as a pipeline of two subtasks. The first part is a text segmentation task and is covered in chapter 4. The second and main part is the prediction of the discursive repertoire for a single text span. The decision to split the task into two different subtasks rather than classifying the text directly in a sequence of repertoires must be considered a design choice. This choice is justified by the presence in the dataset of the annotations of the two subtasks. Dividing the task, it is possible to have two problems simpler than the original.

Prediction of the discursive repertoire is formalised as a text classification task. Given the Hyperion dataset, the main goal is to create a model capable of predicting a label from a finite set (i.e. the discursive repertoire) taking as input a span from the optimal division of the text. Formally, given a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ where $\boldsymbol{x}_i$ is a sequence of tokens and y_i is an integer number $\{0, \ldots, 22\}$ representing a label assigned to each discursive repertoire in the dataset, the goal is to create a model $y = f(\boldsymbol{x})$ mapping the vector representation of the text span $\boldsymbol{x}$ to a label y.

3.3 Machine learning pipeline

This section covers the entire pipeline to solve the problem starting from raw text. The core model is BERT described in section 2.5, so the preprocessing step and training protocols must be adapted to this architecture.

3.3.1 Preprocessing

In section 2.3.1 different preprocessing techniques are introduced. For this task, the preprocessing phase is quite simple, and it is composed of the following steps:

- *Tokenization:* Due to the BERT architecture, the tokenization process is performed using the WordPiece algorithm, a subword tokenizer described in section 2.5;
- *Lowercasing:* All uppercase characters are converted to lowercase;
- *Truncation and padding:* Texts with length greater than 512 tokens are truncated and shorter texts are padded with the special token *[PAD]*;
- *Hashtag unpacking:* In some experiments, hashtags are unpacked to improve model performance. For example, the hashtag "#PresidenteDelConsiglio" is converted to "Presidente Del Consiglio";
- *Normalization:* URLs, numbers, hours and dates, and other special symbols are converted to special tokens;

Hashtag unpacking and Normalization are not applied to all the experiments.

3.3.2 Architectures

As stated above, the core model tested for this task is BERT. The entire architecture is shown in figure 3.3. After the preprocessing step, the model is composed of a pretrained version of BERT and a classification layer added on top, composed of a fully connected linear layer randomly initialised and a softmax function to produce the output probabilities. The final output is the vector of probabilities related to the [CLS] token and the class with the highest probability is chosen to label the example.

Taking into account that the Hyperion dataset is in Italian, three different versions of BERT are tested:

- Multilingual BERT;
- AlBERTo;
- DBMDZ BERT.

These architectures are described in section 2.5.7. The model structure is the same for all three models, so we can consider them as different weights initialisation for the same model.

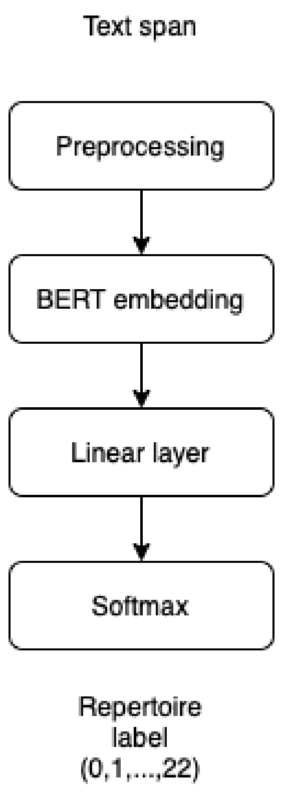


Figure 3.3: pipeline for discursive repertoires classification

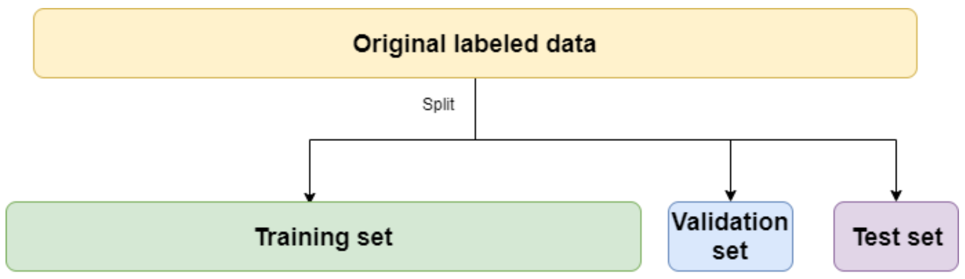


Figure 3.4: From the original dataset to train, validation and test sets

3.3.3 Train, validation and test split

In order to train and evaluate the model, the dataset is split into two parts. 80% as a training set and 20% as a test set, for performance and error analysis. From the training set, another 20% is extracted as a validation set for the selection of hyperparameters, as shown in figure 3.4. Due to the unbalance of classes within the Hyperion dataset, the subset percentage is extracted from each class, in order to obtain the same repertories distribution in train, validation and test set.

3.3.4 Training

As described in section 2.5.1, BERT exploits transfer learning, so its training phase is split into two parts. The pretraining of BERT is very intensive in computation terms, so this work has deepened only the fine-tuning step. After the initialisation of BERT with a pretrained version and a random initialisation of the classifier, the entire model is trained with a standard approach based on the optimization of a loss function.
The loss function chosen is a weighted *categorical cross-entropy*, defined as follows:

$$Loss = -\frac{1}{N}\sum_{i}^{N}\sum_{c}^{C} w_c y_i^{(c)} \log \hat{y}_i^{(c)} \tag{3.1}$$

where $\hat{y}_i$ is the model output of i-th sample, y_i is the target and w_c is a weight assigned to the class to handle the unbalanced dataset. The weights vector $\boldsymbol{w}$ is considered an hyperparameter. Different strategies are tested in the validation test, starting from simply calculating the weight of a class with respect to the others and then rescaling all the values in different ranges. Other techniques are tested, such as the heuristic applied to a logistic regression model described in [16].

The second fundamental component for learning is an optimizer. According to most published works on Tranformer-based architectures, the choice is AdamW optimizer [21]. This algorithm is based on adaptive learning rate and weight decay regularization. During training, the following hyperparameters has been optimized on the validation set:

- Learning rate;
- Batch size;
- Epochs;
- Class weights strategy.

All models have been trained applying the *mixed precision* technique [23]. Mixed precision allows faster training exploiting a combination of single and half-precision floating point representations, obtaining with high probabilities neither a loss of accuracy nor different hyperparameters selection from standard training. The biggest advantage of this technique is a significant reduction in training time and GPU memory. To use this technique, it is necessary to use loss scaling. Loss scaling means multiplying the output of the loss function by some scalar number before performing backpropagation. In this way, it is possible to avoid vanishing of gradients due to truncation in half-precision of them. All models are trained on a single recent GPU and this technique has allowed halving the training time.

3.4 Evaluation

The standard classification metrics are calculated for each tested model, but only some of them are taken as a reference to evaluate performance and select the best model.

- *Accuracy:* It is defined as the number of correctly classified examples divided by the total number of examples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

  Accuracy is a useful metric when classes are balanced in the dataset. This is not the case, so the metric is computed, but it is not used to select the best model and analyze the performance.

- *Precision:* It is defined as the ratio of correct positive predictions to overall positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

  Higher precision for a class means that when a model outputs a prediction of that specific class, probably the example really belongs to that class. For instance, in a classification task of spam/not spam emails, we will look for a model with high precision because we want to avoid the case in which a model predicts a not spam email as a spam and the person does not read the useful email;

- *Recall:* It is defined as the ratio of correct positive predictions over the overall number of positive examples.

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

  Higher recall for a class means that many examples that really belong to that specific class, probably they will be predicted correctly by the model. For instance, in a classification task of people with cancer/not cancer we will look for a model with high recall because we want to avoid the case in which a person with cancer is classified wrongly, so he will not look for a medical examination.

- *F1 score:* It is defined as the harmonic mean of precision and recall.

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.5}$$

  This metric is useful when we care about both precision and recall. The F1 score is a specific instance of $F\beta$ score where the parameter $\beta = 1$, so precision and recall impact the metric equally. Higher values of β mean more importance to recall with respect to precision.

Precision, recall and F1 return a numerical value between 0 and 1 to evaluate performances of the model for each class. In order to obtain an evaluation of performances in the entire multiclass problem, it is necessary to calculate the average of all the classes evaluation. There are three common strategies to compute the average:

- *Macro average:* It is the arithmetical mean of a metric computed for each class. This strategy considers each class equally important for the final computation. In the case of the Hyperion dataset it is useful because it allows to evaluate the performances of less frequent classes, since they are weighted equally to highly frequent ones;

- *Micro average:* The metric is computed as an aggregation of terms for each class, so the true positives are the sum of true positives of all classes. Other terms composing the formulas are computed equally. In this case all the examples contribute equally to the final metric, so classes containing more examples are more important. The main problem of micro average in multiclass problems is that micro F1 = micro precision = micro recall = accuracy, so the metrics are less meaningful;

- *Weighted average:* It is calculated applying weights for each class during the average computation. Like the micro average, all examples contribute equally, but in this case, disadvantages of micro average are avoided. In the experiments this metrics is used to provide an overall evaluation of the models, since it weights classes according to their distribution in the dataset.

3.5 Experiments

All the three pretrained versions of BERT architecture described in section 3.3.2 are tested during experiments. In order to improve performances, different inputs are tested, such as providing BERT a pair of sentences containing the spans and its context. The context of a span is defined as a triplet composed of the left adjacent, the current, and the right adjacent span. This strategy has not improved the performance of the model, so each of the following results is obtained providing only the text span as input.

The table 3.3 shows the main aggregate metrics computed on the test set for the three models with the hyperparameters optimized on the validation set. The most important metrics chosen to evaluate these models are the weighed F1 score and the macro F1 score. DBMDZ-BERT seems to perform better in both, and this is a quite surprising result, due to the nature of AlBERTo that is pretrained on a dataset retrieved on Twitter, so more similar to Hyperion dataset.

Weighted F1 score returns an overall evaluation of all classes, so DBMDZ-BERT obtains better performances in relation to the unbalance of the dataset. Instead if we want to evaluate performances of less frequent classes, macro F1 score is a good indicator due to the fact that less frequent classes weight

Table 3.3: Performance of BERT classifiers for discursive repertoires prediction on the test set

	M-BERT	**DBMDZ-BERT**	**AlBERTo**
Weighted F1	0.33	**0.36**	0.35
Macro F1	0.23	**0.27**	0.26
Accuracy	0.33	**0.37**	0.34
Weighted Precision	0.35	0.37	**0.38**
Weighted recall	0.32	**0.37**	0.35

equally of most frequent. Only taking into account the weighted precision, AlBERTo performs better than DBMDZ-BERT.

Figure 3.5 and 3.6 show how the loss function and the weighted F1 score of DBMDZ-BERT vary during training. After 3 epochs, the model tends to overfit and both loss and metric diverge on the validation set with a longer training phase. Regarding how the DBMDZ-BERT performs on single classes,

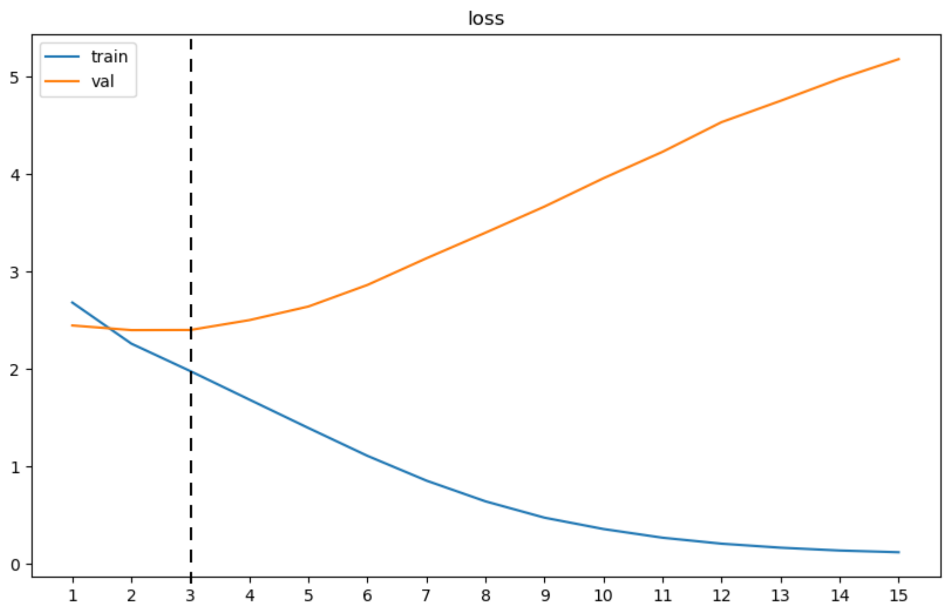


Figure 3.5: Loss function plot for 15 epochs training. Model selected is trained 3 epochs

it is possible to see that there are some repertories with more than 0.6 in prediction accuracy, such as Description and Prescription. Instead, as expected, less frequent classes, such as Anticipation, Confirmation, and Generalization, show poor performance, although weighted loss.

Another problem of the model that is visible in both table 3.4 and figure 3.7 is that DBMDZ-BERT is overpredicting the Certify reality class. This is the most frequent class in the dataset, so to deal with this problem a weighted loss is used, but for this specific repertoire the problem persists. In conclusion, DBMDZ-BERT is chosen as a model for repertoire prediction for downstream tasks described in chapter 6.

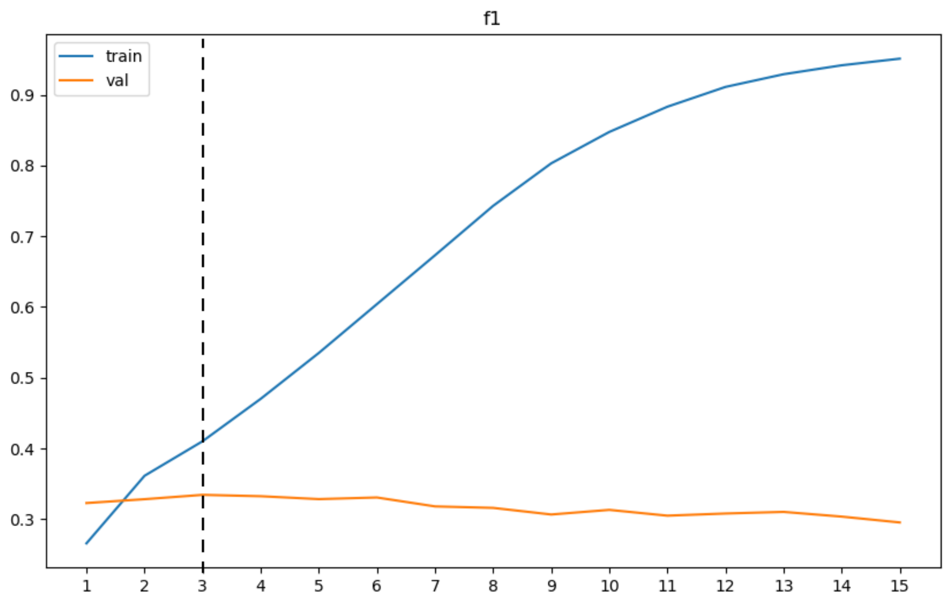
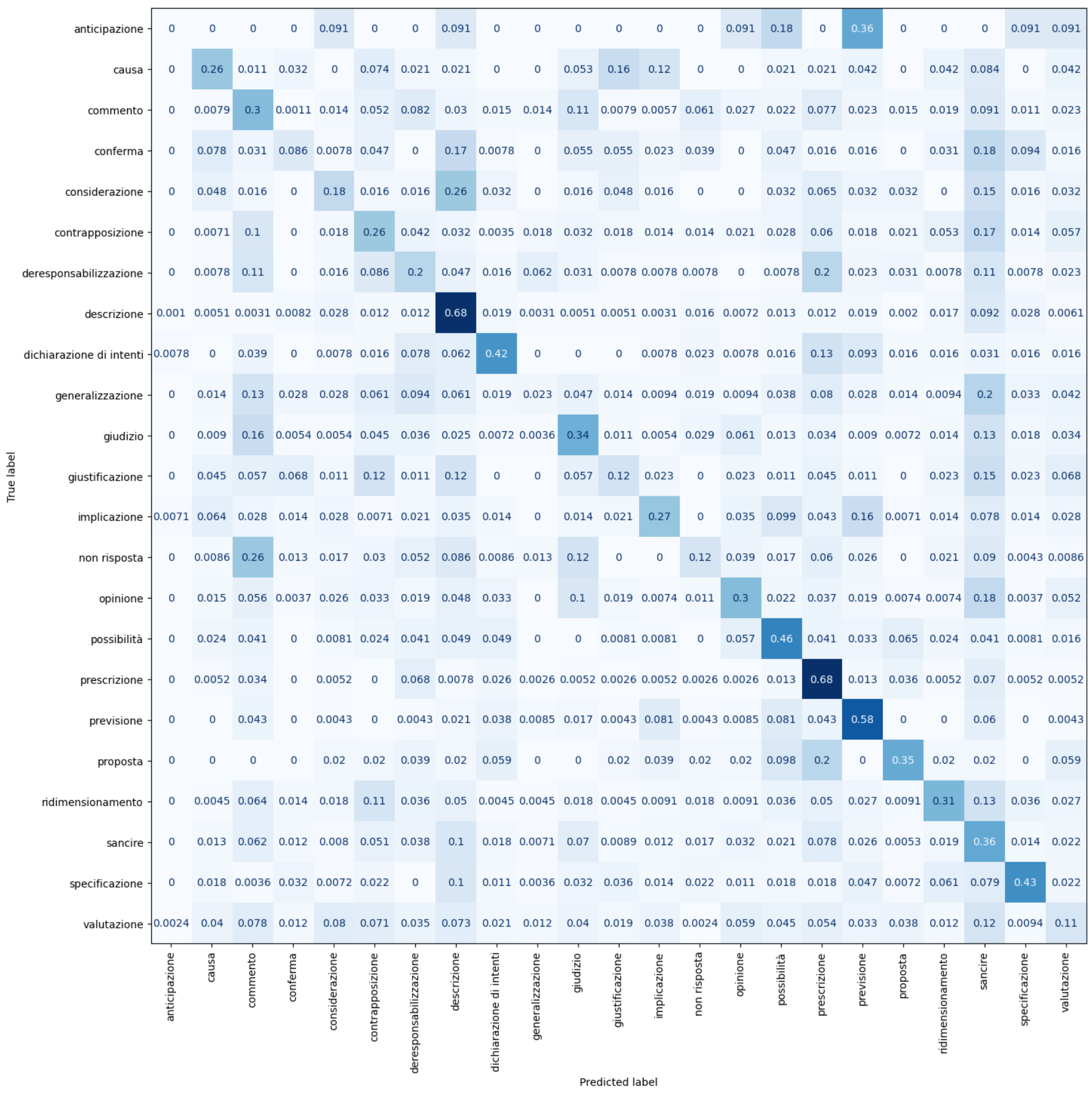


Figure 3.6: Weighted F1 score variations plot for 15 epochs training. Model selected is trained 3 epochs



	anticipazione	causa	commento	conferma	considerazione	contrapposizione	deresponsabilizzazione	descrizione	dichiarazione di intenti	generalizzazione	giudizio	giustificazione	implicazione	non risposta	opinione	possibilità	prescrizione	previsione	proposta	ridimensionamento	sancire	specificazione	valutazione
anticipazione	0	0	0	0	0.091	0	0	0.091	0	0	0	0	0	0	0.091	0.18	0	0.36	0	0	0	0.091	0.091
causa	0	0.26	0.011	0.032	0	0.074	0.021	0.021	0	0	0.053	0.16	0.12	0	0	0.021	0.021	0.042	0	0.042	0.084	0	0.042
commento	0	0.0079	0.3	0.0011	0.014	0.052	0.082	0.03	0.015	0.014	0.11	0.0079	0.0057	0.061	0.027	0.022	0.077	0.023	0.015	0.019	0.091	0.011	0.023
conferma	0	0.078	0.031	0.086	0.0078	0.047	0	0.17	0.0078	0	0.055	0.055	0.023	0.039	0	0.047	0.016	0.016	0	0.031	0.18	0.094	0.016
considerazione	0	0.048	0.016	0	0.18	0.016	0.016	0.26	0.032	0	0.016	0.048	0.016	0	0	0.032	0.065	0.032	0.032	0	0.15	0.016	0.032
contrapposizione	0	0.0071	0.1	0	0.018	0.26	0.042	0.032	0.0035	0.018	0.032	0.018	0.014	0.014	0.021	0.028	0.06	0.018	0.021	0.053	0.17	0.014	0.057
deresponsabilizzazione	0	0.0078	0.11	0	0.016	0.086	0.2	0.047	0.016	0.062	0.031	0.0078	0.0078	0.0078	0	0.0078	0.2	0.023	0.031	0.0078	0.11	0.0078	0.023
descrizione	0.001	0.0051	0.0031	0.0082	0.028	0.012	0.012	0.68	0.019	0.0031	0.0051	0.0051	0.0031	0.016	0.0072	0.013	0.012	0.019	0.002	0.017	0.092	0.028	0.0061
dichiarazione di intenti	0.0078	0	0.039	0	0.0078	0.016	0.078	0.062	0.42	0	0	0	0.0078	0.023	0.0078	0.016	0.13	0.093	0.016	0.016	0.031	0.016	0.016
generalizzazione	0	0.014	0.13	0.028	0.028	0.061	0.094	0.061	0.019	0.023	0.047	0.014	0.0094	0.019	0.0094	0.038	0.08	0.028	0.014	0.0094	0.2	0.033	0.042
giudizio	0	0.009	0.16	0.0054	0.0054	0.045	0.036	0.025	0.0072	0.0036	0.34	0.011	0.0054	0.029	0.061	0.013	0.034	0.009	0.0072	0.014	0.13	0.018	0.034
giustificazione	0	0.045	0.057	0.068	0.011	0.12	0.011	0.12	0	0	0.057	0.12	0.023	0	0.023	0.011	0.045	0.011	0	0.023	0.15	0.023	0.068
implicazione	0.0071	0.064	0.028	0.014	0.028	0.0071	0.021	0.035	0.014	0	0.014	0.021	0.27	0	0.035	0.099	0.043	0.16	0.0071	0.014	0.078	0.014	0.028
non risposta	0	0.0086	0.26	0.013	0.017	0.03	0.052	0.086	0.0086	0.013	0.12	0	0	0.12	0.039	0.017	0.06	0.026	0	0.021	0.09	0.0043	0.0086
opinione	0	0.015	0.056	0.0037	0.026	0.033	0.019	0.048	0.033	0	0.1	0.019	0.0074	0.011	0.3	0.022	0.037	0.019	0.0074	0.0074	0.18	0.0037	0.052
possibilità	0	0.024	0.041	0	0.0081	0.024	0.041	0.049	0.049	0	0	0.0081	0.0081	0	0.057	0.46	0.041	0.033	0.065	0.024	0.041	0.0081	0.016
prescrizione	0	0.0052	0.034	0	0.0052	0	0.068	0.0078	0.026	0.0026	0.0052	0.0026	0.0052	0.0026	0.0026	0.013	0.68	0.013	0.036	0.0052	0.07	0.0052	0.0052
previsione	0	0	0.043	0	0.0043	0	0.0043	0.021	0.038	0.0085	0.017	0.0043	0.081	0.0043	0.0085	0.081	0.043	0.58	0	0	0.06	0	0.0043
proposta	0	0	0	0	0.02	0.02	0.039	0.02	0.059	0	0	0.02	0.039	0.02	0.02	0.098	0.2	0	0.35	0.02	0.02	0	0.059
ridimensionamento	0	0.0045	0.064	0.014	0.018	0.11	0.036	0.05	0.0045	0.0045	0.018	0.0045	0.0091	0.018	0.0091	0.036	0.05	0.027	0.0091	0.31	0.13	0.036	0.027
sancire	0	0.013	0.062	0.012	0.008	0.051	0.038	0.1	0.018	0.0071	0.07	0.0089	0.012	0.017	0.032	0.021	0.078	0.026	0.0053	0.019	0.36	0.014	0.022
specificazione	0	0.018	0.0036	0.032	0.0072	0.022	0	0.1	0.011	0.0036	0.032	0.036	0.014	0.022	0.011	0.018	0.018	0.047	0.0072	0.061	0.079	0.43	0.022
valutazione	0.0024	0.04	0.078	0.012	0.08	0.071	0.035	0.073	0.021	0.012	0.04	0.019	0.038	0.0024	0.059	0.045	0.054	0.033	0.038	0.012	0.12	0.0094	0.11

Figure 3.7: Confusion matrix of discursive repertoires predictions

Table 3.4: Most frequent errors in in discursive repertoire prediction with DBMDZ-BERT

Repertoire	**Accuracy**	**Most frequent error**	**Error frequency**
Anticipation	0	Prediction	0.36
Cause of action	0.26	Justification	0.16
Certify reality	0.36	Description	0.1
Comment	0.3	Judgment	0.11
Confirmation	0.08	Certify reality	0.18
Consideration	0.18	Description	0.26
Contraposition	0.26	Certify reality	0.17
Declaration of aim	0.42	Prescription	0.13
Description	0.68	Certify reality	0.09
Evaluation	0.11	Certify reality	0.12
Exemption from responsibility	0.2	Prescription	0.2
Generalization	0.02	Certify reality	0.20
Judgment	0.34	Comment	0.16
Justification	0.12	Certify reality	0.15
Implication	0.27	Prediction	0.16
Non answer	0.12	Comment	0.26
Opinion	0.3	Certify reality	0.18
Possibility	0.46	Proposal	0.06
Prediction	0.58	Implication	0.08
Prescription	0.68	Certify reality	0.07
Proposal	0.35	Prescription	0.2
Reshaping	0.31	Certify reality	0.13
Specification	0.43	Description	0.1

Chapter 4

Text segmentation

In the previous chapter it is used the optimal division of text spans to perform the classification in discursive repertoires.
This chapter covers a deep learning approach to partition a text in a set of text spans. After a brief presentation of the task, the focus will be on how to evaluate this task and compare deep learning models to other approaches provided by famous NLP libraries.

4.1 Task definition

Text segmentation (a.k.a Boundary detection) is the task of splitting text into meaningful segments. Each segment can be split following different rules. We can perform word segmentation, sentence segmentation, or more complex tasks such as topic segmentation in which each span corresponds to a different topic. Division of a text into multiple parts generally is a fairly common task in NLP, because of the need to transform data from raw long text to useful data as input for solving another task. For example, topic segmentation can be useful to perform a text summarization task.

The dialogical science approach starts from raw text and partitions it into spans corresponding to a sequence of repertoires. In this work this task is designed as a pipeline of two subtasks, text segmentation and then span classification, described in chapter 3. It follows that a procedure is necessary to split text into multiple spans.

During years, different techniques are released to solve the text segmentation task:

- *Rules-based approaches:*
  - Regular expressions;
  - Lexycal rules;
  - Heuristic approaches.
- *Machine learning approaches:*
  - Unsupervised;

– Supervised.

In this chapter, a supervised machine learning approach based on the BERT architecture is described. Formally, given a text defined as a sequence of tokens $\boldsymbol{x} = (x_1, \ldots, x_n)$, the goal is to assign a label $l = \{0, 1\}$ to each token, where the value 1 corresponds to a span boundary and 0 otherwise. The result is a sequence of labels $\boldsymbol{y} = (y_1, \ldots, y_n)$.

As stated above, a text span in the context of dialogical science is defined as a piece of text in which the repertoire does not change. So, the main idea is to train a supervised model on Hyperion dataset capable of detecting the boundaries between two spans that represent a change of discursive repertoire. Although the contextual definition of span, empirically it is possible to determine that usually a span corresponds to a sentence or a part of sentence and often the boundaries correspond to punctuation, so the task can be interpreted as a variation of sentence segmentation. Hence, the personal machine learning approach is compared with other approaches that solve the sentence segmentation task, such as:

- *NLTK sentence tokenizer:* Unsupervised algorithm to perform sentence segmentation [17];
- *Trankit:* Multilingual toolkit based on the XLM-Roberta transformer architecture$_{LARGE}$ [26]. This model has a shared body with different layers on top, each of them trained to solve a different problem.

4.2 Machine learning pipeline

This section describes the entire pipeline, based on BERT, to solve the text segmentation task, starting from raw text. In this case, the preprocessing phase also needs to be adapted to the architecture.

4.2.1 Preprocessing

The following preprocessing techniques are applied to the text:

- *Tokenization:* Due to the BERT architecture, the tokenizaztion process is performed using the WordPiece algorithm, a subword tokenizer described in section 2.5. Punctuation, that is really important for this task has a correspondent token for each symbol;
- *Lowercasing:* All the uppercase characters are converted to lowercase;
- *Double punctuation removal:* Multiple punctuation tokens are removed before providing the sequence as input. For example (“no”, “!”,“!”,“!”) becomes (“no”, “!”);
- *Padding:* Sequences of tokens are padded to obtain an equal lenght;

- *Sequence split:* Instead of truncation, long sequences ares split and fed to the model in more than one step.

Hyperion dataset is composed of a text and a set of spans, so it is necessary to compute the sequence of labels. Starting from the set of optimal spans, in order to create a sequence of labels independent from the architecture, the boundaries are represented at word level. So the ground truth is a sequence $y = (y_1, \dots, y_n)$ where each label $l = \{0, 1\}$ is equal to 1 if it is the last word of the span and 0 otherwise. Then, due to the tokenizer used by BERT, labels are converted to match the subword tokenization.
In some cases, there were examples not correctly annotated, with the ground truth spans that were not a partition of the text. These examples are cleaned reconstructing the text as a union of all optimal spans.

4.2.2 Architecture

The BERT-based architecture is similar to the one for discursive repertoire prediction described in section 3.3.2. The only difference is on the top layer. As shown in figure 4.1, the classification layer is applied on all the tokens and not only to the $[CLS]$ token. Hence, the output of the model is a sequence of labels.
Once the output is returned, the labels are propagated from subword level to

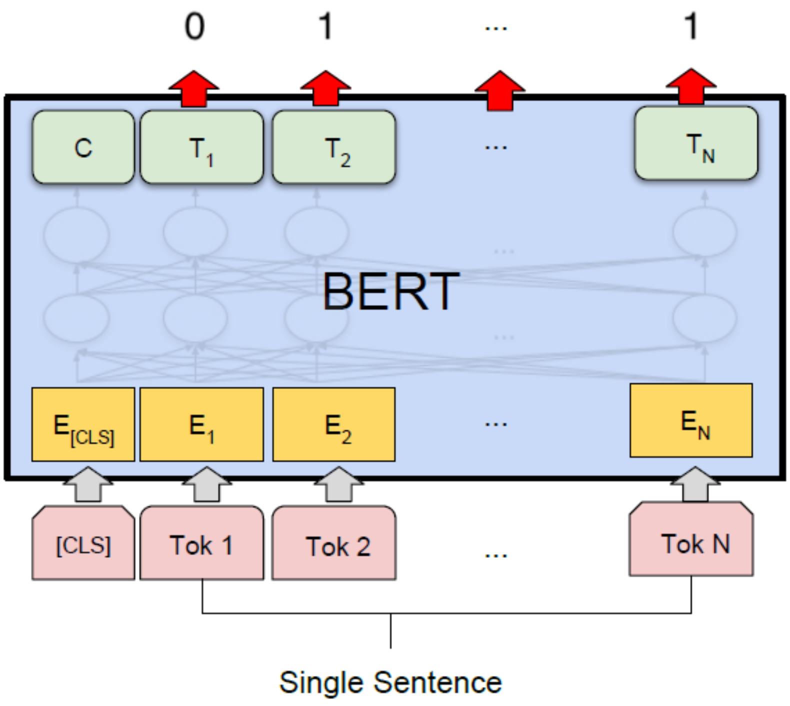


Figure 4.1: Bert architecture for token classification to perform text segmentation

word level, shifting the labels that represent a boundary to the last subword token of a word. For example, if the sequence of tokens is (“basta”,“con”, "questo", "corona", "##virus") and the prediction is (00010) will be converted to (00001), since the double hashtag indicates a subword token.

4.2.3 Training

The starting point is a pretrained version of BERT. All the three architectures described in 2.5.7 are tested. Similarly to BERT for the prediction of discursive repertoires, the classifier on top of BERT is randomly initialized. In this case we perform a binary classification task so the loss function is the binary simplified definition of cross-entropy, but it is used in its weighted version, defined as follows:

$$Loss = -\frac{1}{N}\sum_{i}^{N} w_1(y_i \log \hat{y}_i) + ((1-y_i)\log(1-\hat{y}_i)) \tag{4.1}$$

where $\hat{y}_i$ is the model output of i-th sample, y_i is the target, and w_1 is the weight assigned to the positive class (the boundary) to handle the imbalance between the tokens that are boundaries and normal ones. The weight w_1 is considered a hyperparameter. During the computation of loss functions, all special tokens are masked and the cross-entropy is not computed on these. Also in this case the optimizer remains AdamW and the training phase is computed with the mixed precision approach.

4.2.4 Postprocessing

To compute the segmentation, we divide the text assuming that after this operation the spans will be classified into discursive repertoires. From the theory, each span is defined as a piece of text in which the repertoire does not change. Hence, after the segmentation it is possible to exploit the classification labels to join adjacent segments with the same discursive repertoire, as shown in figure 4.2. This postprocessing step and its performance improvements are analyzed in section 4.4.

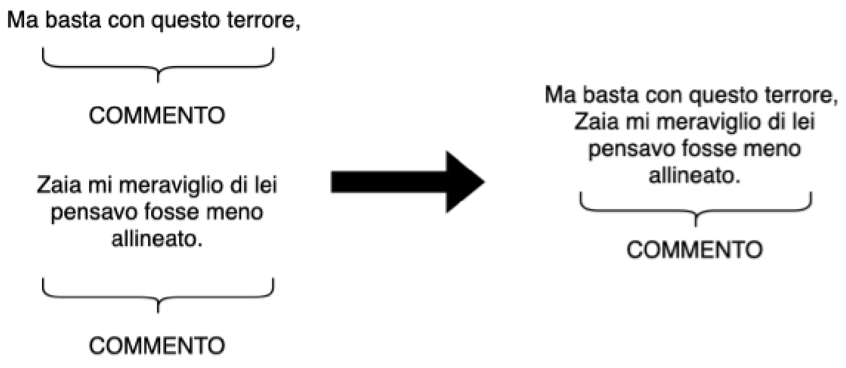

Ma basta con questo terrore,
COMMENTO
Zaia mi meraviglio di lei pensavo fosse meno allineato.
COMMENTO
Ma basta con questo terrore, Zaia mi meraviglio di lei pensavo fosse meno allineato.
COMMENTO

Figure 4.2: Example of spans joined together

It must be noted that the effectiveness of this procedure is strictly dependent on performance of the discursive repertoire classifier. To analyze how much can powerful this procedure can be, some experiments are done also with the gold labels.

4.2.5 Train, validation and test split

Similarly to what described in section 3.3.3, the dataset is split in two parts. 80% as a training set and 20% as a test set, for performance and error analysis. From the training set, another 20% is extracted as a validation set for the selection of hyperparameters.

Due to the low mean number of spans for text (2.3) and the high percentage of non-split texts (63%) in Hyperion dataset, validation set and test set are created in two different versions. The first is full, instead the second is a subset containing only texts split in more than one span. The subsets allows to bypass some problems during the validation phase, described below in section 4.4.

4.3 Evaluation

Although this is a classification task, it is possible to show that the common metrics used to evaluate classification tasks are not effective in this case. Let $y = (00010001)$ be the optimal division of a text, this means that a text is composed of two spans four words long. Let $\hat{y}_1 = (00001001)$ and $\hat{y}_2 = (00000001)$ two possible predictions of the model. Clearly $\hat{y}_1$ is a better solution with respect to $\hat{y}_2$ because the division in two spans is detected but on a wrong token. This kind of error is considered a partial error, instead not detecting a boundary in considered a serious error.

Common classification metrics evaluation are based on the number of correct labels, so precision, recall, and F1 score return higher values on $\hat{y}_2$ with respect to $\hat{y}_1$. With this metrics is impossible to detect partial errors, moreover they are evaluated worse than serious errors. For this reasons it is necessary to look for metrics capable of distinguishing between these two types of errors.

In this work, two metrics are chosen for the text segmentation evaluation:

- *Intersection over union:* It is defined as the intersection of two spans divided by the union of the two spans:

$$IoU = \frac{\text{Sequence of Overlap}}{\text{Sequence of Union}} \tag{4.2}$$

  This metric is derived from Object detection task in computer vision, where the IoU is applied on matrices representing the position of a specific object inside an image. In this case, it can be interpreted as a variation in which intersection and union are computed on a one-dimensional vector of tokens, as shown i figure 4.3.

  The metric is computed between a predicted span and the best match from the ground truth;

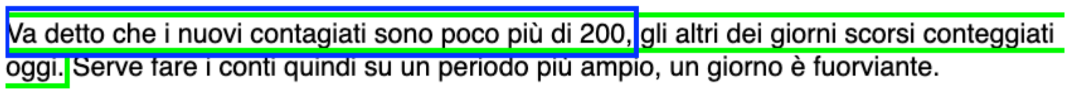

Va detto che i nuovi contagiati sono poco più di 200, gli altri dei giorni scorsi conteggiati oggi. Serve fare i conti quindi su un periodo più ampio, un giorno è fuorviante.

Figure 4.3: Example of intersection over union applied on the first predicted span

- *Generalized Hamming distance* [6]: It is a metric that extends the concept of traditional Hamming distance to give partial credit for near misses. Hamming distance measures the minimum number of symbols to change to transform a string in the other one. In this case, the two strings are the sequences of binary labels.

  To evaluate partial errors, this metric introduces another operation, the shift, which allows to swap two adjacent labels and has a lower penalty than insertion and deletion. For example, if a substitution is evaluated with a coefficient equal to 1, and shift is equal to 0.5, a two-position shift has a lower cost. Therefore, in a case similar to the one described above, the partial correct prediction $\hat{y}_1$ will be better evaluated than the completely wrong prediction $\hat{y}_2$. A variation of the shift coefficient can make this operation more or less cheaper than substitution, so it is possible to change the size of what we consider a partial error. In this work, the cost of substitution operation is set to the mean length of a span and the cost of shift is set to 1.

One of the main differences between the two metrics is that the generalized Hamming distance evaluates the segmentation of a text, instead intersection over union evaluates a single span. Hence, in IoU, it is necessary to compute the average to obtain an evaluation of how well a text is segmented.

4.4 Experiments

All experiments are divided into two categories. The first includes all the experiments done with ready-to-use toolkits and libraries, such as NLTK and Trankit described in section 4.1. These two approaches are mixed with the use of regular expressions and the performances of both of them are evaluated. For clarity of writing, only the best result obtained with this approach is described, which consists of the NLTK sentence tokenizer without using regular expressions.

The second category regards the personal approaches based on BERT. In this task, the initialization of BERT with one of the three pretrained versions covered in section 2.5.7 is considered a hyperparameter selection, so the chosen version is the one with the highest performance in the validation set. Therefore, when referring to a general BERT, a model initalized with DBMDZ-BERT is intended, which is the one with the best performance in the validation set.

During the experiments a key problem is encountered for BERT fine-tuning. The selection of hyperparameters, especially the selection of loss weights, is affected by the high percentage of single-span texts, that is, 63%. In section 4.2.5 is stated that two versions of validation set are generated. The second is a subset of the validation set containing only texts split in more than one span. Table 4.1,shows the main problem, that is a bad selection of the loss weight w_1 corresponding to the boundary label. A selection of low values for w_1 implies bad learning, because the model is not punished enough when misleads a boundary and it is basically a model that prefers not to split the text. The output of this model will almost always be a single span. In fact, on the full validation set, there are 2400 examples that in the best model are divided into 2570 spans, while the optimal division is a total of 5770 spans. Of course, this selection is strictly influenced by the high number of single-text span in Hyperion dataset, so as a conclusion, it is not useful to select hyperparameters on this validation set.

Table 4.1: Performances of BERT model for segmentation for different weights of the loss function tested in the validation set

Weight	**Segments**	**IoU**	**GHD**
20	7986	0.44	82
10	6064	0.53	65
5	4783	0.59	51
3	2801	0.73	39
1	2570	**0.76**	**25**

To solve this problem, the weight is selected using a subset of the validation set composed of only texts split in more than one span. As shown in table 4.2, now the model with the best performance in terms of metrics has selected the weight $w_1 = 5$. In addition, the number of predicted spans is more similar to the ground truth in this subset of the validation set, meaning that the model is splitting the text. How good the split is is evaluated by the two metrics.

Table 4.2: Performances of BERT model for segmentation for different weights of the loss function tested in the subset of texts split in more than one span

Weight	**Segments**	**IoU**	**GHD**
20	7986	0.48	58
10	6064	0.54	49
5	4783	0.60	**45**
3	2801	**0.63**	49
1	2570	0.56	60

After optimization of the BERT architecture, its performance is compared to the approach based on NLTK sentence tokenizer and the naive approach to directly not splitting the text. Moreover, the postprocessing technique based on joining adjacent spans with the same assigned repertoire is evaluated. Table 4.3 shows the final results considering three postprocessing modalities:

- No postprocessing;

- Postprocessing using labels predicted by the model for discursive repertoire prediction;
- Postprocessing using ground truth labels.

In this way, it is possible to evaluate if the postprocessing phase is useful and how it is influenced by the performance of the model. The results in table 4.3

Table 4.3: Performances of segmentation models in the full test set with and without postprocessing

Model	**Postprocessing**	**IoU**	**GHD**
FULL TEXT	-	**0**.83	29
NLTK	-	0.38	67
NLTK	BERT	0.41	59
NLTK	GT	0.87	27
BERT	-	0.42	55
BERT	BERT	**0.47**	**46**
BERT	GT	0.91	11

show that the BERT approach improves performance with respect to NLTK. Also, the postprocessing step guarantees a performance improvement, but it is strictly dependent from the classifier performance. Ideally, with an optimal classifier, it is possible to maximize the metrics. However, the classifier trained and described in the previous chapter causes a slight improvement of general performances.

Once again it is necessary to take into account the high percentage of single-span texts. The computation of metrics without splitting the text allows them to be maximized, obtaining $IoU = 0.83$, but this result is clearly influenced by the features of the Hyperion dataset. So in order to evaluate these approaches and obtain more general results, models are evaluated on a subset as well containing non-single-span texts.

Table 4.4 shows the results on the subset of test set composed only of non-single-span texts.

Table 4.4: Performances of segmentation models in the non-single-span subset of test set, with and without postprocessing

Model	**Postprocessing**	**IoU**	**GHD**
FULL TEXT	-	0.23	82
NLTK	-	0.50	69
NLTK	BERT	0.53	56
NLTK	GT	0.88	27
BERT	-	0.65	41
BERT	BERT	**0.67**	**40**
BERT	GT	0.84	31

In both test sets, it is possible to notice that BERT outperforms NLTK, with or without the union as a postprocessing step. In the subset, the spread between BERT and NLTK increases, so BERT performs far better than NLTK in non-single-span texts. Analyzing the differences between the two metrics, it seems that the spread between NLTK and BERT is wider considering GHD, especially in table 4.3. This fact is interpreted by taking into account the coefficients assigned to operations in GHD. Recalling that the shift operation is penalized 20 times less with respect to insertion and elimination, this metric tends to punish a lot the cases in which the model prediction contains a different number of spans from the ground truth. Otherwise, if the number of spans is correct, the metric is lower because the algorithm simply shifts a character rather than a substitution.

As a conclusion, BERT is chosen as a segmentation model for future work described in the next chapters, but the NLTK sentence tokenizer performances are not so distant from a fine-tuned model, so in relation to the effort of usage and the great advantage in terms of computational speed must be taken into consideration.

Chapter 5

Pipeline evaluation

This chapter focuses on the evaluation of the entire pipeline for discursive repertoires prediction. Starting from raw text, the aim is to construct a pipeline composed of the two models described in chapters 3 and 4 and to evaluate them to solve the overall problem. In the first section, methods and metrics for the evaluation are covered. The chapter then focuses on the final results obtained and the analysis of the importance of each step that makes up the pipeline.

5.1 Evaluation metrics

The whole problem seen as a unique task is defined as the prediction of a sequence of discursive repertoires, starting from a text. Formally, given a text representation $\boldsymbol{x} = (x_1, \ldots, x_n)$ the model $\boldsymbol{y} = f(\boldsymbol{x})$ returns a sequence of labels $\boldsymbol{y} = (y_1, \ldots, y_m)$ where $y_i = \{0, \ldots, 22\}$. This task has many similarities with the more famous sequence labeling ones, such as named entity recognition. One of the main differences is that labels form a partition of the text. In fact, each word belongs to a repertoire and cannot be outside, so there is no "outside" label.

The main goal of this evaluation is to measure the impact of both the segmentation step and the classification step. The general idea is to evaluate the classification taking into account how good the segmentation is, assigning partial values to correctly classified repertoires that do not fully match the gold annotation span. Researchers have already published work on similar tasks, such as [25] for the named entity recognition task and [10] for fine-grained analysis of propaganda. In the second work, a variation of precision, recall and F1 score is defined to handle partial overlaps.

Let s be a classified span and t be a span in the gold annotation. The following function returns an evaluation of these two elements:

$$C(s, t, h) = \frac{|(s \cap t)|}{h} \delta(l(s), l(t)) \tag{5.1}$$

where h is a normalization factor and δ is the evaluation function of the classification in discursive repertoires. In this case, all repertoires are considered equally distant from the others, so $\delta(l(s), l(t)) = 1$ if $l(s) = l(t)$ and 0 otherwise. Future implementations of the repertoires classifier could be evaluated in a different way, such as taking into account the dialogical weight. Therefore, the function δ could be converted to a distance function between classes that evaluates how dissimilar the two repertoires are, as suggested by the table of repertoires in figure 2.1.

The function C is then used to define the three metrics as follow:

$$P(S,T) = \frac{1}{|S|} \sum_{\substack{s\in S \\ t\in T}} C(s,t,|s|) \tag{5.2}$$

$$R(S,T) = \frac{1}{|T|} \sum_{\substack{s\in S \\ t\in T}} C(s,t,|t|) \tag{5.3}$$

where S is the predicted text partition in spans and T is the optimal division. The precision and recall defined above can be used to compute the F1 score with its common formula.

The term $|S|$ in 5.4 and $|T|$ in 5.3 have the task of penalizing the evaluation when the segmentation step produces too or few spans. However, due to the nature of the task that implies a partition of the text, there are some edge cases that are not well evaluated by these metrics. In figure 5.1 is shown an example of two possible predictions, S and S' and the ground truth T. The colour indicates a different class. In this case, S has precision $P = 0.5$ and recall $R = 0.54$. Instead S' has $P = 0.33$ and $R = 0.54$. Clearly S' is a better

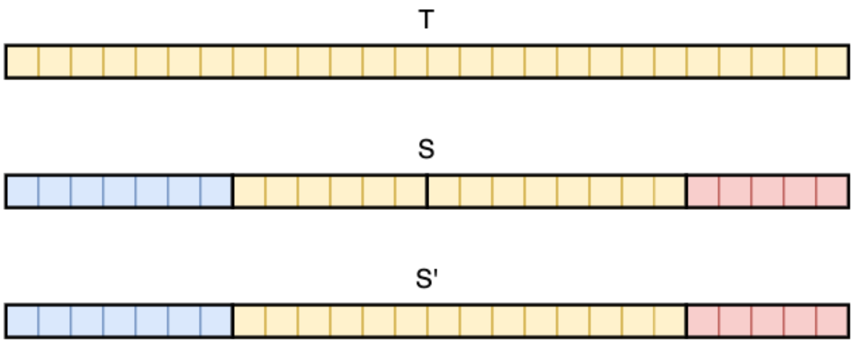


Figure 5.1: Two examples of different predictions.

solution, but it is evaluated worse than S in both precision and F1 score, maintaining the same recall. The example just described is the case when the postprocessing step joins spans with the same predicted repertoire.

In this thesis, a slight variation of these metrics is introduced to overcome the problem covered above and handle tasks in which text is partitioned:

$$IoU(S,T) = \frac{1}{|S|} \sum_{\substack{s\in S \\ t\in T}} C(s,t,|(s\cup t)|) \tag{5.4}$$

It is defined equally with respect to precision, but the normalization factor h is equal to the length of the union between s and t. The metric is called

Intersection over Union because the function C literally becomes the IoU multiplied by the function δ, which is the result of classification. Returning to the previous example S has $IoU = 0.13$, instead S' has $IoU = 18$.

5.2 Experiments

In order to evaluate the pipeline on unseen data, Hyperion dataset is split into a training set composed of 80% and a test set with the remaining 20%. The training set is used to retrain models described in chapters 3 and 4. Hence, the next results are reported by computing metrics on a 20% of data not seen in all the models used. To show a general view of the performances of this pipeline, different combinations of models are analyzed. The segmentation step is processed in the following three ways:

- NLTK sentence tokenizer;
- BERT for text segmentation;
- Optimal segmentation.

Optimal segmentation is added to experiments in order to establish an upper bound on the influence of text segmentation step.

The classification in discursive repertories is handled with the following classifiers:

- BERT for discursive repertoire classification;
- Random classifier;
- Optimal classifier.

In addition to BERT, the pipeline is evaluated with a random classifier to set the lower bound and analyze how much the second step can influence the metrics and the pipeline performances. For the same reason, an optimal classifier is tested, returning labels from the ground truth.

In table 5.1 the results of all experiments are reported. Excluding optimal and random models, the best pipeline is composed of two BERT and the postprocessing step is applied. This is an expected result, since it is a composition of the best approaches when analyzed independently from the others.

The segmentation phase influences the pipeline evaluation, in fact, it is possible to pass from IoU equal to 0.21 to 0.27 with BERT and postprocessing. Its effectiveness is demonstrated by the optimal segmentation, in fact, with the same repertoires classifier it is possible to arrive at 0.42 of IoU, optimizing the segmentation process. Another confirmed result is that the classification step is the most important. With the random classifier, on 23 categories the metrics are close to 0, and with the optimal one, precision, recall and F1 score are close to 1. The spread is so large because of the definition of δ term in the metrics functions, that returns 1 when the prediction is correct and

Table 5.1: Pipeline evaluation results

Pipeline	**Postprocessing**	**Precision**	**Recall**	**F1**	**IoU**
NLTK + BERT	-	0.30	0.30	0.29	0.21
NLTK + BERT	$\surd$	0.29	0.30	0.28	0.24
BERT + BERT	-	**0.32**	0.30	0.30	0.25
BERT + BERT	$\surd$	0.31	**0.30**	**0.30**	**0.27**
OPT SEG + BERT	-	0.42	0.42	0.42	0.42
OPT SEG + BERT	$\surd$	0.42	0.42	0.42	0.42
NLTK + CLS	-	0.95	0.88	0.90	0.62
NLTK + OPT CLS	$\surd$	0.94	0.88	0.90	0.94
BERT + OPT CLS	-	0.95	0.87	0.89	0.67
BERT + OPT CLS	$\surd$	0.95	0.87	0.89	0.93
NLTK + RAND CLS	-	0.04	0.04	0.04	0.03
NLTK + RAND CLS	$\surd$	0.04	0.04	0.04	0.03
BERT + RAND CLS	-	0.04	0.04	0.04	0.03
BERT + RAND CLS	$\surd$	0.04	0.04	0.04	0.03

0 otherwise. Probably, an implementation of a distance metric to evaluate predicted repertoires would reduce the spread between the random and the optimal classifier and would reduce the impact of this step with respect to the entire pipeline.

With the optimal classifier and without postprocessing IoU is not much higher than the results obtained with BERT. The reason is the definition of this metric that has the normalization factor $h = |s \cup t|$, so it is the most sensitive metric to the segmentation step. Good results for IoU can only be achieved with good segmentation.

As shown in the example in section 5.1, precision recall and F1 score are not able to deal with the union of segments, in fact, a general loss of performance is noticeable when the postprocessing is applied. Regarding postprocessing, as already stated in chapter 4, it is strictly dependent on the classifier. As shown by IoU values, better performance has the classifier and greater will be the increment using postprocessing.

Chapter 6

Downstream NLP tasks

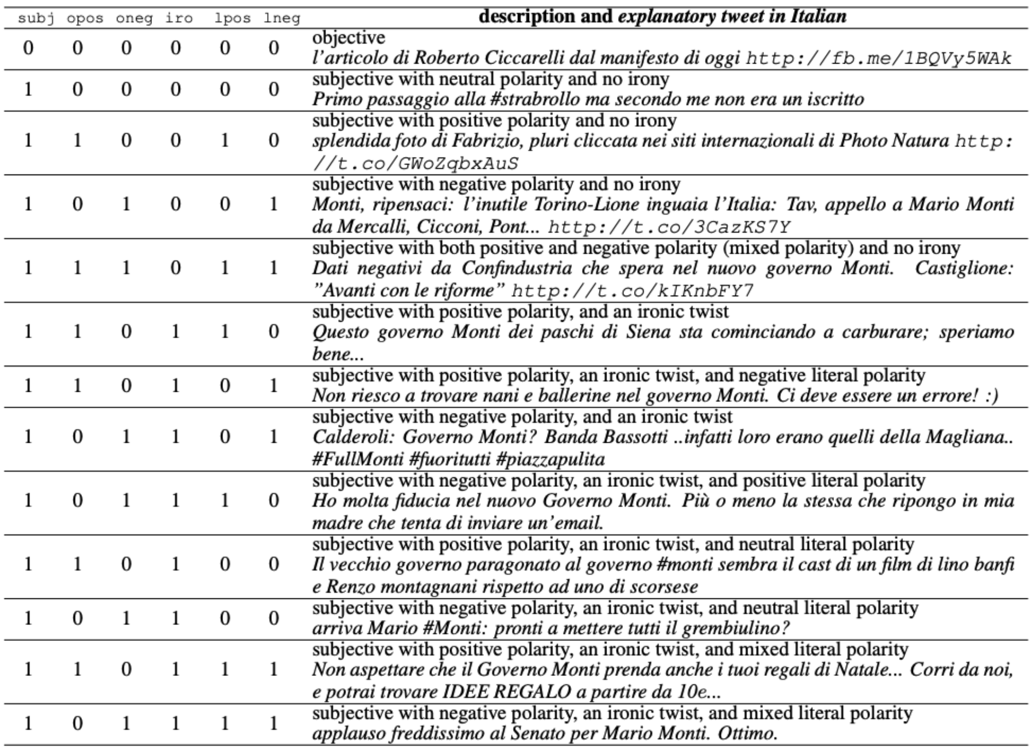
This chapter covers the use of discursive repertoires as features for solving common NLP tasks. In particular, the focus is on the Evalita 2016 SENTIPOLC competition, which is composed of three different tasks: subjectivity classification, polarity classification and irony detection. The first section covers the competition and a brief analysis of the dataset. Then, the focus is on the three tasks and the machine learning approach based on the prediction of discursive repertoires through the models previously described.

6.1 SENTIPOLC dataset

SENTIment POLarity Classification Task [4], is a challenge proposed in 2014 and then in 2016 based on a dataset composed of Twitter messages in Italian language. As shown in figure 6.1, the dataset set includes 7410 tweets with different annotations:

- *Subjectivity (subj):* Binary label indicating whether a message is subjective or objective;
- *Overall positive (opos)*: Binary label indicating the overall positive sentiment;
- *Overall negative (oneg)*: Binary label indicating the overall negative sentiment;
- *Literal positive (lpos)*: Binary label indicating a literal positive sentiment;
- *Literal negative (lneg)*: Binary label indicating a literal positive sentiment;
- *Ironic*: Binary label indicating the presence of irony.

The literal sentiment is available as additional information for the tasks. It provides the real sentiment ignoring the figurative usage of the language. Hence some tweets can have an inversion of polarity between literal and overall sentiment. This is the case of ironic tweets.



subj	opos	oneg	iro	lpos	lneg	**description and *explanatory tweet in Italian***
0	0	0	0	0	0	objective *l'articolo di Roberto Ciccarelli dal manifesto di oggi* `http://fb.me/1BQVy5WAk`
1	0	0	0	0	0	subjective with neutral polarity and no irony *Primo passaggio alla #strabrollo ma secondo me non era un iscritto*
1	1	0	0	1	0	subjective with positive polarity and no irony *splendida foto di Fabrizio, pluri cliccata nei siti internazionali di Photo Natura* `http://t.co/GWoZqbxAuS`
1	0	1	0	0	1	subjective with negative polarity and no irony *Monti, ripensaci: l'inutile Torino-Lione inguaia l'Italia: Tav, appello a Mario Monti da Mercalli, Cicconi, Pont...* `http://t.co/3CazKS7Y`
1	1	1	0	1	1	subjective with both positive and negative polarity (mixed polarity) and no irony *Dati negativi da Confindustria che spera nel nuovo governo Monti. Castiglione: "Avanti con le riforme"* `http://t.co/kIKnbFY7`
1	1	0	1	1	0	subjective with positive polarity, and an ironic twist *Questo governo Monti dei paschi di Siena sta cominciando a carburare; speriamo bene...*
1	1	0	1	0	1	subjective with positive polarity, an ironic twist, and negative literal polarity *Non riesco a trovare nani e ballerine nel governo Monti. Ci deve essere un errore! :)*
1	0	1	1	0	1	subjective with negative polarity, and an ironic twist *Calderoli: Governo Monti? Banda Bassotti ..infatti loro erano quelli della Magliana.. #FullMonti #fuoritutti #piazzapulita*
1	0	1	1	1	0	subjective with negative polarity, an ironic twist, and positive literal polarity *Ho molta fiducia nel nuovo Governo Monti. Più o meno la stessa che ripongo in mia madre che tenta di inviare un'email.*
1	1	0	1	0	0	subjective with positive polarity, an ironic twist, and neutral literal polarity *Il vecchio governo paragonato al governo #monti sembra il cast di un film di lino banfi e Renzo montagnani rispetto ad uno di scorsese*
1	0	1	1	0	0	subjective with negative polarity, an ironic twist, and neutral literal polarity *arriva Mario #Monti: pronti a mettere tutti il grembiulino?*
1	1	0	1	1	1	subjective with positive polarity, an ironic twist, and mixed literal polarity *Non aspettare che il Governo Monti prenda anche i tuoi regali di Natale... Corri da noi, e potrai trovare IDEE REGALO a partire da 10e...*
1	0	1	1	1	1	subjective with negative polarity, an ironic twist, and mixed literal polarity *applauso freddissimo al Senato per Mario Monti. Ottimo.*

Figure 6.1: SENTIPOLC dataset

One of the key features of this dataset is that labels and tasks depend on the others. For example, a tweet is considered objective if and only if $opos = oneg = lpos = lneg = iro = 0$. Another example is the irony class that must be subjective and have the inversion of polarity described above. For this dependency between labels, some published approaches use *multi-task learning*.

The competition is evaluated on a hidden test set that includes 2000 Twitter messages.

6.2 Tasks

The approach based on discursive repertoires is tested in all three tasks of the original 2016 competition. In order to make a comparison between the already published technique and the one discussed in this thesis, all models are tested with the metrics chosen by the organizers on the original test set.
Task and metrics used are the following:

- *Subjectivity Classification:* Given a text, a model must predict whether the text is subjective or not. This is a binary classification task, and it is evaluated on precision, recall and F1 score. The reference metric is the macro average of the F1 score.
- *Polarity classification:* Given a text, a model must predict whether a text has neutral, positive, negative or mixed sentiment. This task is composed

of four different labels, which are all the combinations of the binary labels describing the overall sentiment. According to the competition rules, overall positive and overall negative labels are predicted by two independent models. Hence, this task is divided into two different binary classification tasks.

Metrics for evaluation are precision recall and f1 score. Then, the final evaluation is the mean of the two macro averaged F1 scores:

$$\begin{aligned} F^{pos} &= \frac{(F_0^{pos} + F_1^{pos})}{2} \\ F^{neg} &= \frac{(F_0^{neg} + F_1^{neg})}{2} \\ F &= \frac{(F^{neg} + F^{neg})}{2} \end{aligned} \tag{6.1}$$

- *Irony detection:* Given a text, a model must predict whether a text is ironic or not. Similarly to subjectivity classification, it is a binary classification task evaluated on a macro-averaged F1 score.

6.3 Machine learning pipeline

The main goal of these experiments is to understand whether it is possible to extract useful information from discursive repertoires. The SENTIPOLC dataset includes Twitter messages and three different problems to be solved. The pipeline presented in this section is the same for all three tasks.

The figure 6.2, shows the steps applied to solve the tasks. The main component is a support vector machine, described in section 2.6, with the purpose of taking input information received by the discursive repertoire prediction.

Starting from the raw text, the first step is to predict the repertoires that belong to the text. To do this, the approach is to follow the steps of the best pipeline chosen in chapter 5. Hence, the raw text is preprocessed with the same techniques used for the text segmentation task and then the text is split by the already trained BERT. From the text segmentation step, multiple spans can be returned in output. Then, each span is fed as input to the classifier.

At this point, multiple strategies for feature extraction are tested. The first group of strategies is to simply extract discursive repertories and construct a 23-dimensional vector based on their frequencies, in two modalities:

- *Absolute frequencies:* Each scalar in the vector represents the count of a repertoire;
- *Relative frequencies:* Each scalar represents the count of a repertoire divided by the number of spans. The vector sum is 1.

This simple strategy allows us to create low-dimensional vectors, but due to the short length of Twitter texts, vectors are sparse and not very full of information about the input example. For this reason, more complex techniques are tested, based on the extraction of BERT embeddings covered in Section 2.5.6

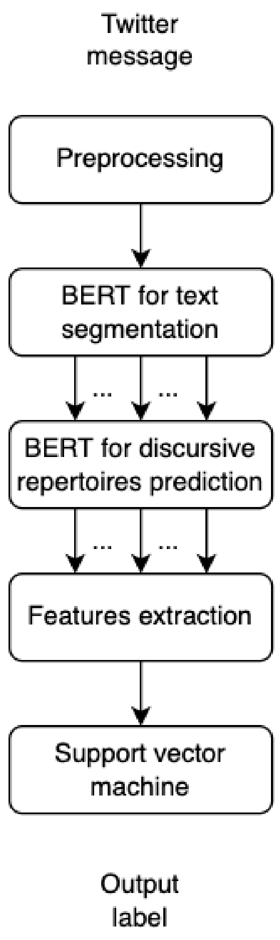


Figure 6.2: Pipeline for SENTIPOLC tasks based on discursive repertoires

- *BERT last layer embeddings:* The output of the last BERT enconder (last layer) is extracted. It is a matrix with dimensions equal to the number of tokens and the size of the hidden states. So, in BERT$_{BASE}$ is (512,768). Then these embeddings are averaged to return a 768-dimensional vector
- *BERT last four layers embeddings concatenation:* based on the test by the original BERT authors. This is the best way to perform the contextual embeddings extraction. The output of the last four BERT encoders is concatenated, returning a matrix of dimension (512, 3072). Then, all the tokens are averaged, and the final vector is 3072-dimensional.
- *BERT last layer [CLS] token embedding:* As stated in section 2.5, the $[CLS]$ special token is added and then used to generate the label of a sequence classification problem. During training, the BERT attention mechanism applied to this token focuses on highlighting the most important relations to solve the task. Then the embedding of this token is the input for the final layer that classifies the sequence. Hence, this token is interpreted similarly to a learned sentence embedding without the need to average or somehow combine token embeddings.

To handle the possibility of multiple spans in output from the segmentation step, BERT embeddings are extracted for each span and then averaged.

After the extraction step, a support vector machine is trained. The training set is divided into 5 folds. Hyperparameters are selected using a grid search, so each combination of hyperparameters is tested using cross-validation and taking the F1 score as a reference to select the best model. Then, the model is retrained on the entire training set using the best hyperparameters selection.

6.4 Experiments

Similar experiments are carried out for the three tasks. The main goals of these experiments are:

- Find which repertoires allow to distinguish example of the classes;
- Find the best strategy to provide information about discursive repertoires to a machine learning model;
- Check performances of this pipeline with respect to the already published results.

The next section provides an overview of results obtained for every single task in the SENTIPOLC challenge.

6.4.1 Subjectivity Classification

Subjectivity classification is defined as a binary classification task in which a model must predict whether a tweet is subjective or not. In SENTIPOLC training set 5098 examples are annotated as subjective and 2312 are objective. The first step is to predict the discursive repertoires in the training set. In figure 6.3 the predictions are grouped by class and normalized.

An exploratory analysis shows that some repertoires appear to be correlated with subjective class, such as “Comment”, “Judgment”, “Opinion” and “Prediction”. Instead, “Description”, “Exemption from responsibility”, “Non answer”, “Reshaping” and “Certify Reality” are correlated with objective class.

Then, all the strategies described are tested to solve the task. In Table 6.1 it is possible to see all results. The selected strategy is the hidden state of

Table 6.1: Performance of SVM for subjectivity classification with all the feature extraction strategies

	F1 subj	**F1 obj**	**F1 Macro**
Absolute fr	0.60	0.62	0.61
Relative fr	0.61	0.60	0.60
Last layer	**0.83**	0.58	0.71
4 last layers concat	0.80	0.72	0.76
[CLS] last layer	0.80	**0.72**	**0.76**

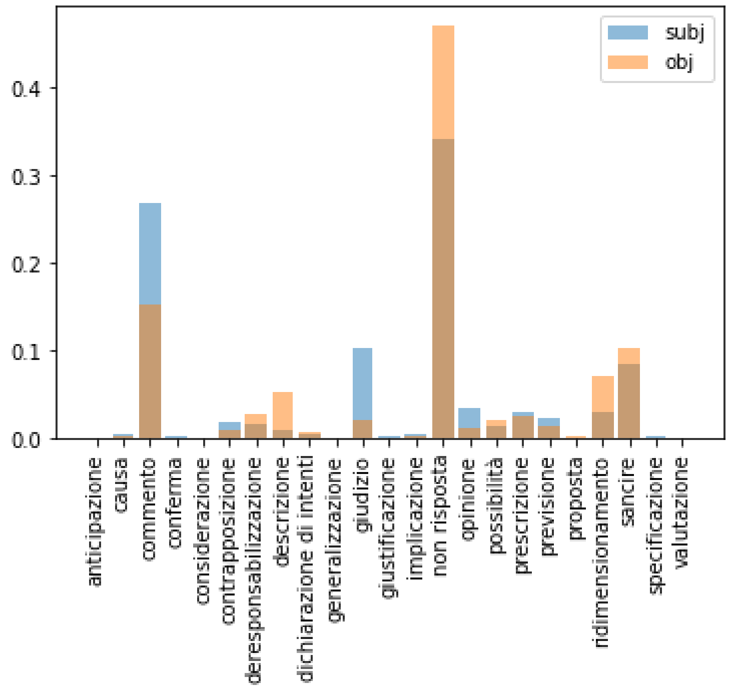


Figure 6.3: Discursive repertoires distribution grouped by subjective and objective classes

the last layer on $[CLS]$ token, The 4 layers concatenation strategy has similar performance but the vector representation is bigger so the other is preferred. As expected, it is possible to notice a performance difference between the embedding strategy and methods based on frequencies of discursive repertoires. However, these techniques are interesting because they do not provide any information on the text, but only on discursive repertoires.

The table 6.2 shows the best selected model, called DR-SVM, compared to the work already published in 2016 during the competition and the best model published in the following years. The SVM approach based on discursive

Table 6.2: Ranking on subjectivity classification task

	F1 subj	**F1 obj**	**F1 Macro**
AlBERTo [31]	**0.84**	**0.75**	**0.79**
DR-SVM	0.80	0.72	0.76
Unitor.1.u	0.81	0.68	0.74
Unitor.2.u	0.80	0.67	0.74
samskara.1.c	0.78	0.66	0.72

repertoires is close to AlBERTo, the version trained on Italian tweets described in section 2.5.7. The information extracted by the repertoires allows to do reach the second best score. In this case, AlBERTo fine-tuned on the task still outperforms our approach based on discursive repertoires.

6.4.2 Polarity classification

As described above, the polarity classification task is defined as a set of two independent tasks and then their metrics are averaged together to obtain a unique performance evaluation.

In the SENTIPOLC dataset there are 2051 overall positive tweets and 5359 overall non positive tweets. Regarding negative sentiment, 2983 messages are overall negatives and 4427 are overall not negatives.
As described above, labels are dependent, indeed we can consider the objective class equal to the class of neutral (i.e $opos = 0$ and $oneg = 0$) tweets. So, the plot in figure 6.3 can also be interpreted as the repertoires grouped by polarizing tweets and neutral ones.

The same features extraction techniques are tested for this task, with results in table 6.3, taking into account the two independent tasks. Hence, the macro average for the positive classification, the macro average for the negative and their average are shown.

Table 6.3: Performance of SVM for polarity classification with all the feature extraction strategies

	F1 Pos	**F1 Neg**	**F1 Avg**
Absolute fr	0.55	0.68	0.62
Relative fr	0.54	0.67	0.61
Last layer	**0.72**	0.76	0.74
4 last layers concat	0.72	0.76	0.74
[CLS] last layer	0.71	**0.77**	**0.74**

The results show that all feature extraction techniques based on BERT hidden states perform quite similarly. Also in this case, the simplest technique is chosen, so the input of the SVM is the hidden state of [CLS] token. Table 6.4 shows the ranking for the polarity classification task. In this specific task many studies on different architectures have been published after the competition.

Table 6.4: Ranking of polarity classification task

	F1 Pos	**F1 Neg**	**F1 Avg**
BERT-Twitter-pp [32]	**0.74**	0.76	**0.75**
DR-SVM	0.71	**0.77**	0.74
AlBERTo [31]	0.72	0.73	0.72
LSTM [20]	0.66	0.74	0.69
CNN [11]	0.65	0.71	0.68
UniPI.2.c	0.68	0.64	0.66

The SVM approach based on discursive repertoires performs appreciably well with the first place on the negative tweets classification, but a work with BERT and a specific preprocessing pipeline for Twitter messages remains the state of the art for this task. Both BERT-Twitter-pp and AlBERTo present a preprocessing pipeline specifically developed for tweets, instead the pipeline described in this thesis uses a quite general preprocessing phase. Based on these studies, future work could implement similar preprocessing steps to improve performance.

6.4.3 Irony detection

In irony detection, the goal is to predict whether a Twitter message is ironic or not. The SENTIPOLC dataset for irony detection is very unbalanced, in fact, there are only 868 tweets labeled as ironic. The other class is composed of 6542 examples.

Due to dependency between labels, ironic tweets are a subset of subjective ones with an inversion of polarity from literal to overall polarity. The dependencies can be observed in figure 6.4, that shows a similar plot to the one in subjectivity classification task, with some differences. Less repertoires appear to be useful to distinguish the two classes, such as "Comment", "Judgment" and "Prediction", which are correlated with the ironic class. Instead, "Description", "Non answer" and "Reshaping" are correlated with non-ironic class.

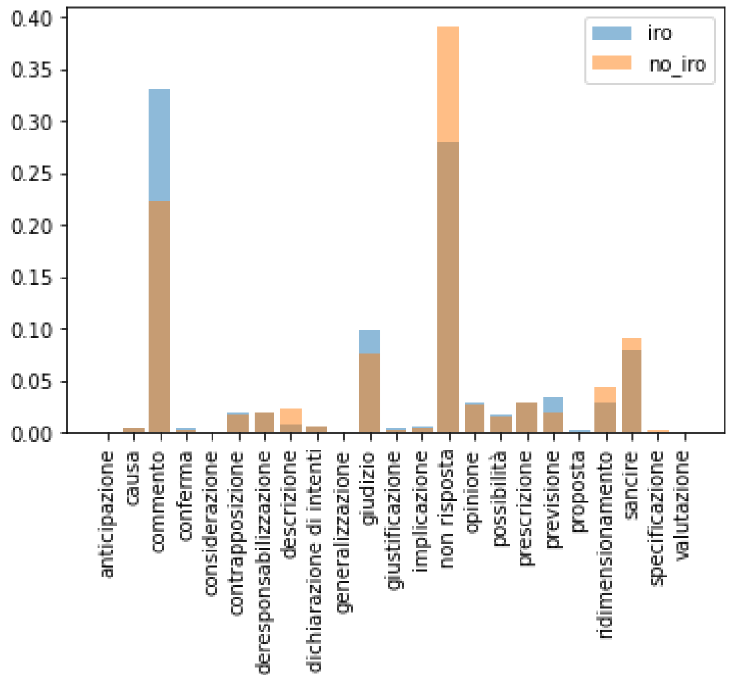


Figure 6.4: Discursive repertoires distribution grouped by ironic and not ironic classes

To select the best feature extraction procedure, once again an SVM with cross-validation and grid search is trained. The table 6.5 shows all the results obtained.

According to F1 score of both classes, the embedding of the $[CLS]$ token is the best input for the SVM. It is also possible to notice that this task appears

Table 6.5: Performance of SVM for irony detection with all the feature extraction strategies

	F1 Iro	**F1 Non-iro**	**F1 Macro**
Absolute fr	0.29	0.88	0.58
Relative fr	0.27	0.87	0.56
Last layer	0.41	0.92	0.66
4 last layers concat	040	0.91	0.65
[CLS] last layer	**0.42**	**0.93**	**0.67**

as the most difficult task with respect to the other two already covered. The table 6.6 shows the final ranking for this task and all the models have lower performances compare to subjectivity classification and polarity classification.

Table 6.6: Ranking of Irony detection task

	F1 Iro	**F1 Non-iro**	**F1 Macro**
DR-SVM	**0.42**	0.93	**0.67**
LSTM [20]	-	-	0.62
AlBERTo [31]	0.28	**0.94**	0.61
CNN [11]	-	-	0.54
tweet2check16.c	0.17	0.91	0.54

The approach based on discursive repertoires increases significantly the performance of irony detection task, obtaining the state of the art for this sub-task. The highest performance spread is in the ironic class, with a 0.14 improvement in the f1 score compared to the AlBERTo model. Furthermore, the input based on repertoire frequencies performs better than the models presented at the challenge in 2016, despite its sparse representation and low dimensionality.

In conclusion, discursive repertoires appear quite effective in detecting irony. As shown above, some repertoires are very frequent on ironic tweets, but this result can be influenced by the performance of the classifier for the discursive repertoires prediction described above. Hence, a future improvement of BERT-based model can make more clear the usefulness of discursive repertoires in natural language processing tasks.

Chapter 7

Conclusions

The main goal of this thesis was to explore the applications of dialogical science and its theory in NLP tasks. In order to reach the goal, the first step is an automated detection of the discursive repertoires.

One of the contributions is a critical analysis of the metrics for evaluating the subtasks and the entire pipeline. The aim was to find a suitable metric that would correctly evaluate partial errors. It is shown that current metrics do not effective return a sensible evaluation in a number of specific cases and it is proposed a novel formulation able to also handle those.

Then, with the aim to automate the dialogical science methodology, it is proposed an approach based on deep learning techniques, designed as a pipeline composed of two subtasks to solve, that is, the text segmentation and the successive prediction of discursive repertoires. The evaluation of the pipeline shows that this subtask is the most important and although the absence of a baseline, we can consider the results obtained as significant. In fact, 23 classes are not easily handled by a machine learning model, especially when the dataset is strongly unbalanced. The most frequent repertoires are detected correctly with more than 60% of accuracy, maintaining the capacity to detect the others sufficiently high in most of cases.

The text segmentation step is a standard task for NLP, so the expectation was a similar performance of the already trained tools for text segmentation. However, the direct and supervised approach of fine-tuning BERT on Hyperion dataset brought about 24% of improvements with respect to the best ready-to-use tool. Furthermore, experiments on the postprocessing phase showed relevant results considering that it is strictly dependent on the classifier performance.

The last contribution is a demonstration of the effectiveness of discursive repertoires applied to downstream tasks. The main idea was to compare an approach based on discursive repertoires to the state of the art techniques. In order to make this comparison, the SENTIPOLC challenge is chosen, because it offers a good variety of tasks and during the years many different approaches to compare have been published. Experiments with discursive repertoires as features for downstream tasks in this challenge are positive, especially considering the fact that the performance of the repertoire classifier could, in some cases, introduce noise in the modelling of the downstream task.

Two representations are tested, a simple one based on the frequencies of the repertoires and one based on the latest layers of the deep network used to identify the repertoires. The inputs based on repertoires frequencies showed that they contain useful information for solving the three tasks we experimented on. Moreover, this approach is easy to interpret, providing the list of repertoires able to discriminate the classes of the downstream task. The representation based on the network layers has been used as input to a Support Vector Machine outperforming the state of the art on the irony detection task. This approach has increased the average F1 score of 0.05 and has increased of 50% the F1 score in the single ironic class with respect to a fine-tuned AlBERTo. In polarity detection, the approach described has obtained second place in the rankings showing performances lower than only 0.01 on the average F1 score with respect to BERT with the application of a complex preprocessing pipeline implemented for this task. In subjectivity classification, the SVM model has reached second place, but with a larger distance from the state of the art. An interesting approach for future work could be a combination between feature of discursive repertoires and feature from a fine-tuned model directly on the downstream task.

Returning to the main goal written in the introduction, this thesis provided empirical evidence that the dialogical science and discursive repertoires extract useful information for tackling NLP tasks. In addition, the pipeline enables to speed up the work of the linguists that manually analyze texts with the dialogical science.

7.1 Future work

The nature of this project is experimental and exploratory, so it provides the first results about this topic and there are many ways to continue the project development. While experiments were carried out, new alternative approaches and future expansions to the current work emerged:

- *English language extension:* Due to the dataset available, this work is based on Italian language. A future work could explore dialogical science applications for English language, in order to verify its effectiveness on a large number of tasks and datasets. Moreover, more recent pretrained models published only for English language could be used to implement the pipeline;

- *Distance metric between repertoires:* In this thesis, discursive repertoires are treated as independent and equally distant classes, but we know from the theory that there are repertoires more similar to some and different from others. Hence, future work could explore this part of theory, exploit that when training the models by weighting the errors according to the distance between the repertoires;

- *Generalized model for repertoires prediction:* It is possible to work on a generalized model for discursive repertoires prediction, joining different datasets already existent to Hyperion dataset. The main advantages is the possibility to exploit other parts of the theory, such as the question, that can cause a different classification for the same text. It is not already implemented due to the presence of only one question in Hyperion dataset. Another advantage is to handle the imbalance of Hyperion dataset adding more examples.

- *Different task definition:* The task of predicting repertoires can be reformulated in a different way. For instance, it can be interpreted as a unique task without splitting it in two subtasks;

- *Different models for downstream tasks:* The SVMs implemented for downstream tasks have produced good performances, but this thesis does not cover alternatives to this model. A future work can focus on the implementation of a neural network for tackling downstream tasks;

- *Combined inputs for downstream tasks:* This thesis is focused on the use of discursive repertoires as main input for solving downstream tasks. An alternative approach is to use them as additional features combined with some others. An idea is to concatenate extracted vectors of BERT for repertoires prediction with a Tranformer-based architecture fine-tuned directly for the downstream task, such as AlBERTo. In this way, it is possible to verify whether information relative to discursive repertoires is complementary to the one retrieved by a fine-tuned tranformer architecture.

This list is not exhaustive, but it can represent a valid starting point for future research.

Bibliography

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[2] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. *CoRR*, abs/2202.03555, 2022.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[4] Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, and Viviana Patti. *Overview of the Evalita 2016 SENTIment POLarity Classification Task*, pages 146–155. 01 2016.

[5] Valerio Basile, Mirko Lai, and Manuela Sanguinetti. Long-term social media data collection at the university of turin. pages 40–45, 01 2018.

[6] Abraham Bookstein, Vladimir Kulyukin, and Timo Raita. Generalized hamming distance. *Information Retrieval*, 5, 10 2002.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[8] A. Burkov. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.

[9] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[10] Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods*

in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 5636–5646, Hong Kong, China, November 2019. Association for Computational Linguistics.

[11] Jan Deriu and Mark Cieliebak. *Sentiment Analysis using Convolutional Neural Networks with Multi-Task Training and Distant Supervision on Italian Tweets*, pages 184–188. 01 2016.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[14] Robert Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *Proceedings of the international conference on Neural Networks*, volume 3, pages 11–14. IEEE Press New York, NY, USA, 1987.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[16] Gary King and Langche Zeng. Logistic regression in rare events data. *Political Analysis*, 9:137 – 163, 2001.

[17] Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.

[18] Turchi G.P. Orrù L. *Metodologia per l'analisi dei dati informatizzati testuali: fondamenti di teoria della misura per la scienza dialogica. Edises.* 2014.

[19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[20] Felice Dell'Orletta Lorenzo De Mattei, Andrea Cimino. Multi-task learning in deep neural network for sentiment polarity and irony classification.

[21] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[22] Hrushikesh N. Mhaskar, Qianli Liao, and Tomaso A. Poggio. Learning real and boolean functions: When is deep better than shallow. *CoRR*, abs/1603.00988, 2016.

[23] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[25] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30, 08 2007.

[26] Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2021.

[27] Luisa Orrù, Christian Moro, Marco Cuccarini, Monia Paita, Marta Silvia Dalla Riva, Davide Bassi, Giovanni Da San Martino, Nicolò Navarin, and Gian Piero Turchi. Machine learning and madit methodology for the fake news identification: the persuasion index. 2022.

[28] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[29] Eleonora Pinto, Alessandro Fabbian, Rita Alfieri, Anna Da Roit, Salvatore Marano, Genny Mattara, Pierluigi Pilati, Carlo Castoro, Marco Cavarzan, Marta Silvia Dalla Riva, Luisa Orrù, and Gian Piero Turchi. Critical competences for the management of post-operative course in patients with digestive tract cancer: The contribution of madit methodology for a nine-month longitudinal study. *Behavioral Sciences*, 12(4), 2022.

[30] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics.

[31] Marco Polignano, Pierpaolo Basile, Marco de Gemmis, Giovanni Semeraro, and Valerio Basile. AlBERTo: Italian BERT Language Understanding Model for NLP Challenging Tasks Based on Tweets. In *Proceedings of the Sixth Italian Conference on Computational Linguistics (CLiC-it 2019)*, volume 2481. CEUR, 2019.

[32] Marco Pota, Mirko Ventura, Rosario Catelli, and Massimo Esposito. An effective bert-based pipeline for twitter sentiment analysis: A case study in italian. *Sensors*, 21(1), 2021.

[33] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[34] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[35] José Luis Rojo-Álvarez, Manel Martínez-Ramón, Jordi Muntilde;oz-Mariacute;, and Gustau Camps-Valls. *Support Vector Machine and Kernel Classification Algorithms*, pages 433–502. 2018.

[36] Williams Rumelhart, Hinton. Learning representations by back-propagating errors, 1986.

[37] Stefan Schweter. Italian bert and electra models, November 2020.

[38] Gian Piero Turchi, Marta Silvia Dalla Riva, Caterina Ciloni, Christian Moro, and Luisa Orrù. The interactive management of the sars-cov-2 virus: The social cohesion index, a methodological-operational proposal. *Frontiers in Psychology*, 12, 2021.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[40] Jesse Vig. Visualizing attention in transformer-based language representation models, 2019.

[41] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.