POLITECNICO DI TORINO
Master degree course in Computer Engineering

Master Degree Thesis

Explaining black-box models in the context of Natural Language Processing

EBANO Text

Supervisors
prof. Tania Cerquitelli

Candidates
Salvatore Greco
matricola 241061

Accademic year 2018-2019

Acknowledgements

I would like to thank my supervisor professor Tania Cerquitelli and Francesco Ventura for giving me the opportunity to do this thesis work. I would also like to thank them for teaching me the approach and the research methodology, as well as technical skills, and above all for having transmitted me the passion for this area.

A heartfelt thanks also goes to all people known during this course of study especially for the personal enrichment. Some of these people were like a second family to me. I have also known people from all the world and this allows me to open my mind.

Finally i would like to thank my family for the support, despite the distance, because without them it would not have been possible to face this course of study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the new era introduced by artificial intelligence and automatically decision making systems is always more important to understand how they take decisions in order to trust these models and let humans understand and interpret their decisions. In order to create models more interpretable by humans, it is always more emerging the field of research of *eXplainable Artificial Intelligence*. The analysis of textual information has a prominent role in our modern society for example by analyzing and discarding cv on job applications, or by analyzing the sentiment for marketing purpose, or by understand query on the search engine or again by creating chatbot that make these models in direct contact with the end user. Besides this, *Natural Language Processing* has made great strides with new types of architectures and models that understand and generate languages more and more in an human-like manner. After a careful study of the state-of-the-art of the black-box models explainability, it is emerged that in the natural language processing field there is a lack of these techniques.

The objective of this thesis is to propose a methodology to explain black-box deep neural networks in the context of natural language processing in order to make these models more transparent and more understandable by humans. For this purpose this work proposes *EBANO Text*, a novel explanation framework that provides a *local explanation* of the model by applying a process of perturbation over interpretable features and measuring the impact of the influence that features have on the original prediction.

This paper is organized as follows: in the second chapter are reported all the mainly notions studied about the state-of-the-art with a particular focus

on *Natural Language Processing* and *eXplainable Artificial Intelligence*. In the third chapter instead is explained the proposed methodology and the architecture of *EBANO Text* in order to show how to obtain a local explanation given an input text and a deep neural network. In the fourth chapter all the experimented choices are motivated, some examples of input text along with their local explanations are presented and some global results are reported by analyzing a large corpus of local explanations. Finally, the last chapter highlights the main conclusions of this thesis and proposes some possible future works to improve the framework and the methodology.

Chapter 2

State-of-the-art

In this chapter, firstly will be described the principal notions about *Machine Learning* as general introduction. After is reported the current state-of-the-art of *Natural Language Processing* and sequence models with a special focus on *BERT* that is the current state-of-the-art for this field of machine learning and is also the case of study of this thesis. The section on sequence models traces the evolution process of recurrent neural networks in order to explain and motivate the reason of the deep neural network chosen and its innovative features. Besides this are described also the principal notions about the theory of *explainability* and *eXplainable Artificial Intelligence* and their current state-of-the-art. This part goes into detail on the explainability theory presenting and motivating the reasons why it is necessary. Moreover are showed in detail the different types of explanation that can be obtained and some examples of well known frameworks and techniques at the state-of-the-art.

2.1 Introduction to Machine learning

Nowadays we hear more and more about *Artificial Intelligence*, *Machine Learning*, *Deep Learning*, *Neural Networks* and *Data Science*. But what these terms really means? This first part attempts to explain and clarify these concepts [12].

Artificial Intelligence or *AI* can be defined as the huge number of tools, algorithms or software for making computers or machines behave intelligently. Artificial intelligence is the larger set and contains all the other subsets.

Machine Learning or *ML* instead is a subset of artificial intelligence and its first definition was given by Arthur Samuel in the 1959:

> *"Field of study that gives computers the ability to learn without being explicitly programmed"*[**Arthur Samuel**]

This implies that a machine learning algorithm is not directly programmed by a set of rules or statements to predict the output as traditional programming paradigm. The key concept of machine learning is the *learning* made by a set of input training examples with both input and label that allows the algorithm to automatically learn the rules to predict the most likely label given an input. In general the objective of machine learning is to learn mapping between input and output, this is also called *supervised learning*.

What is instead a *Neural Network* also called *Artificial Neural Network*? A *Neural Network* or *NN* is a subset of machine learning techniques and consists of a set of neurons and weighted connections. The neural network, looking a lot of training example, learns the function that better approximate the mapping between input and output.

Deep Learning or *DL* is the subset of machine learning algorithms which makes use of neural networks to understand the input and output mapping. The terms deep learning and neural network are used interchangeably but more precisely deep learning refers to neural networks that are "deep" and then with a large number of neurons and hidden layers.

Data Science can be defined as the set of techniques and algorithms that attempt to extract knowledge and insight from data.

In summary deep learning is a subset of machine learning that in turn is a subset of artificial intelligence; data science instead is an intersection of all the other fields as shows in figure 2.1.

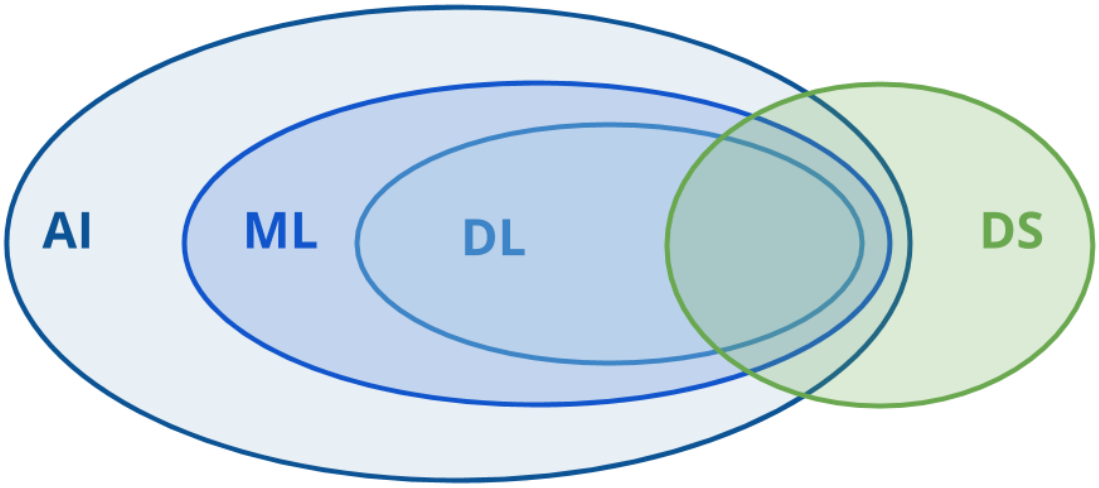

Figure 2.1. Terminology hierarchy.

2.1.1 Types of learning

So far it has been said that the learning phase is the a key concept for machine learning algorithms. There are different existing types of learning :

- *Supervised Learning*: the learning process is allowed by a set of input training examples where are available both input and output labels. The training phase consists of learn the mapping between input and output by looking a large set of training examples. The training, for each input example, is divided in two phases:

  1. *Forward Propagation*: the input X is feeded into the initial layer of the network and is propagated through the hidden layers until the output layer that produces the output Yˆ.
  2. *Backward Propagation*: given the expected output Y and the predicted output Yˆ, the model computes the gradient of the loss function with respect to the weights of the network. The weights are

adjusted proportionally with the gradients calculated and this is the training phase of the model.

Supervised learning is actually the more powerful, more valuable and more established type of learning for neural networks.

- *Unsupervised Learning*: in this case, for the learning phase, are needed only the inputs but not the output labels. The main goal is to find something meaningful or extract new knowledge from the input data. One example of unsupervised learning is the clustering algorithm. Some examples of clustering algorithm are *K-Means*, *DBScan*, *Hierarchical Clustering* etc.
  - The *K-Means* is one of the most widespread and simple clustering algorithm. Starting from some not labeled input data and from the parameter K, that is the number of clusters that want to obtain, the clustering algorithm divides input data in K similar groups. K-means is an iterative process where are selected randomly K centroids, and where each point is assigned to the nearest cluster based on the minimum distance between the point and the centroid. Centroids are recalculated in an iterative way as the average of the points in the cluster. The *K-Means* algorithm needs to know a priori the number of clusters K. This is the main weakness of the algorithm because, specially with high dimensional data, is not easy to know which is the best K. The are some techniques to choose the best number of clusters such as the *Silouhette* calculation that takes into account the *cohesion* inside the clusters and the *division* between different clusters, allowing to run the algorithm with different values of K, taking the one that has a better silhouette score.
- *Reinforcement Learning*: is a semi-supervised learning where an agent, by interacting with the environment, receives a feedback for each action that it takes. The feedback is called reward and can be "good" or "bad". This implies that the goal of reinforcement learning is to maximize this reward. The main problem of this type of learning is that requires a huge amount of data.
- *Transfer Learning*: Given a task A where is available lot of data, it is possible to train a model and gain knowledge from this task and transfer this knowledge to a similar task B where, maybe, is available a smaller dataset. This technique is used a lot in computer vision where, for

example, if training a model for a task A such recognize objects from a big dataset of images is possible to transfer this knowledge to a task B like recognize cats with a smaller dataset and smaller training time. Another example is in the context of natural language processing where, if you learn to represent language as task A, it is possible to transfer this knowledge to a task B like recognize person names in a sentence. This phase of transfer learning from a task A to a task B is also called *fine tuning*.

2.1.2 Types of data

For the training phase of neural networks is needed lot of input data, but which are the different types of data?

- *Structured data*: are classical tabular data where the columns are fixed and each column has a domain. The pattern of this data is fixed and then is relatively easy for a neural network to understand it.

Id	**City**	**Zip Code**	Size (mq)	Parking	#Rooms
House 1	Turin	10129	60	Yes	2
House 2	Milan	20019	50	No	1
House 3	Florence	50100	65	No	2
House 4	Rome	00100	75	Yes	3

Table 2.1. Structured data.

- *Unstructured data*: in this case instead the pattern is not fixed. Classical examples of unstructured data are images, texts, audios and so on. While humans can very easily understand these types of data, for a neural network is more difficult.

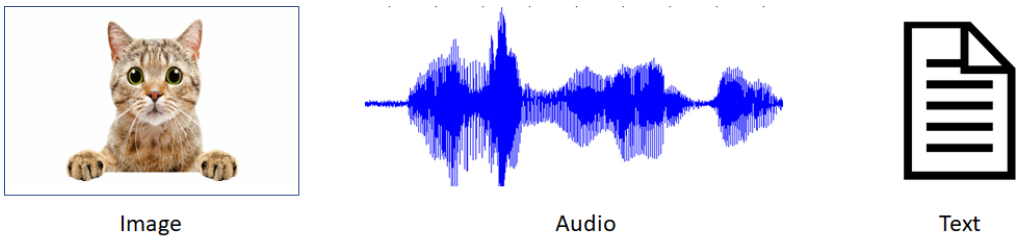

Figure 2.2. Unstructured data.

2.1.3 The exponential growth of machine learning

Machine learning is a more dated field as it is expected, but in the last decades it is exponentially grown. The reasons why machine learning is emerging so much in the last years are the following:

- *Big Data*: in the last years many companies started the digitization phase where always more data is created, saved and then available. Data is fundamental for neural network in particular for the training phase.
- *Performance*: in the last years the performances have grown exponentially, specially with technologies like GPU or TPU that are very suitable for some machine learning tasks.
- *Research and state-of-the-art*: neural networks are becoming more and more powerful thanks to lot of research that now is done by universities and companies. For this reason there are available on internet a lot of open codes and papers.
- *Programming Frameworks*: in the last years have been released some open source software libraries that allow to write neural networks easily and with few lines of code. Using these libraries the programmer can write only some functions to define the model and the training without explicitly programming all the calculations that occurs in the forward and backward propagation making very faster the coding process. Some of the most famous libraries are *Tensorflow*, *PyTorch*, *Caffe*, *Keras* and many others.

2.1.4 Humans and machine learning differences

One fundamental aspect is understanding the difference between the human reasoning from the one of neural networks.

For example, if want an algorithm that given some input features like city, zip code, parking, size, #rooms attempts to predict the price of the house. An human will reasoning by combining some of the inputs to create intermediate features. For example combining city and zip code can express the life quality of the house; combining the presence of parking, the size of the house and the number of rooms can figure out the family friendly of the house and so on. instead, the neural network, in each neuron feeds information from all inputs and combining them predicts the output. The inner features have

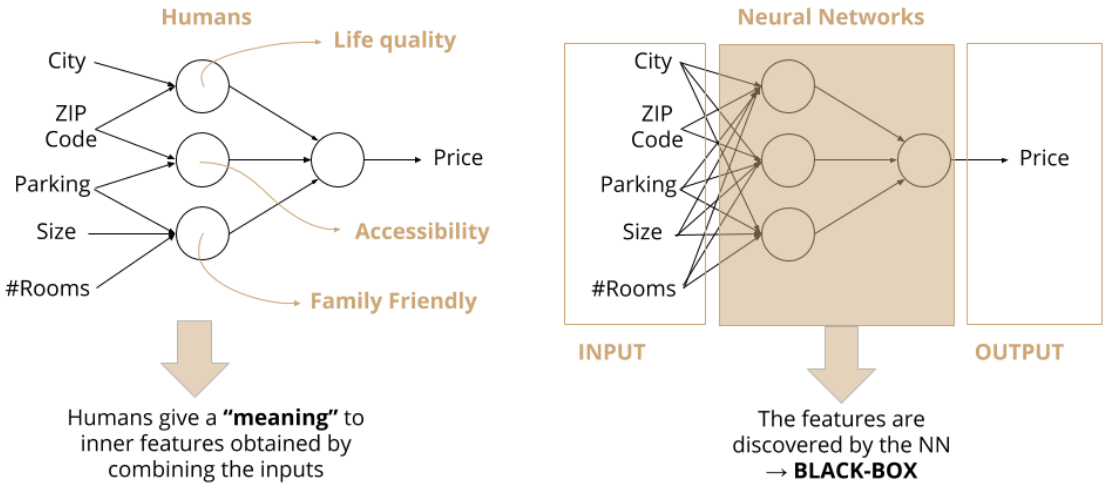

Figure 2.3. Human vs Machine Learning reasoning.

not human understandable meaning but are automatically discovered by the neural network, for this reason it is said that works as a *black-box* and then, given an input, it provides an output and what happens inside it is discovered automatically by the model. The explained example is showed in figure 2.3.

The fact that neural networks work as a black-box, implies that these models are not human understandable. This is one of the main weakness of neural networks nowadays in that, as accuracy they can exceed the human level of performance, but often they cannot be used because is unknown how they take decisions and also their inner behaviour.

2.2 Natural Language Processing

NLP stands for *Natural Language Processing* and is the field of study of the computational treatment of natural language with the main purpose of teaching computers to understand and generate human language. NLP, due to the complex nature of human language, is considered one of the most difficult problem in computer science. It is a very multidisciplinary field which comprises the study of linguistic, statistics and mathematics, computer science, artificial intelligence and so on.

Some examples of typical tasks in NLP are:

- *Sentiment analysis*:
  Consists of analyzing the input text and predicting if the underlying sentiment is positive, negative or neutral. It is also possible to predict a score (for example stars from 1 to 5) of the input text.
- *Topic detection*:
  Classify the input text by assigning a topic tag or a category label that explain the topic of the input.
- *Machine translation*:
  Translate the input text from a language to another language.
- *Named-Entity recognition*:
  Classify each entity of the input text to pre-defined categories of entity such as "person name", "city", "company", "place", "date" and so on. This entity extraction adds knowledge to the input text.
- *Question Answering*:
  Given an input question, try to predict the most likely possible answer.
- *Speech recognition*:
  Given an input audio, the objective is to identify words and phrases in spoken language and convert them to a machine-readable format.
- *Text summarization*:
  Consists of shortening long pieces of input text. The intention is to create a coherent and fluent summary having only the main points outlined in the document.

Two of the main subsets of NLP are:

- NLU: Stands for *Natural Language Understanding* and means understand input text in form of human language to a machine understandable format.

- NLG: Stands for *Natural Language Generation* and means generate output text in form of human language from a machine representation of the language.

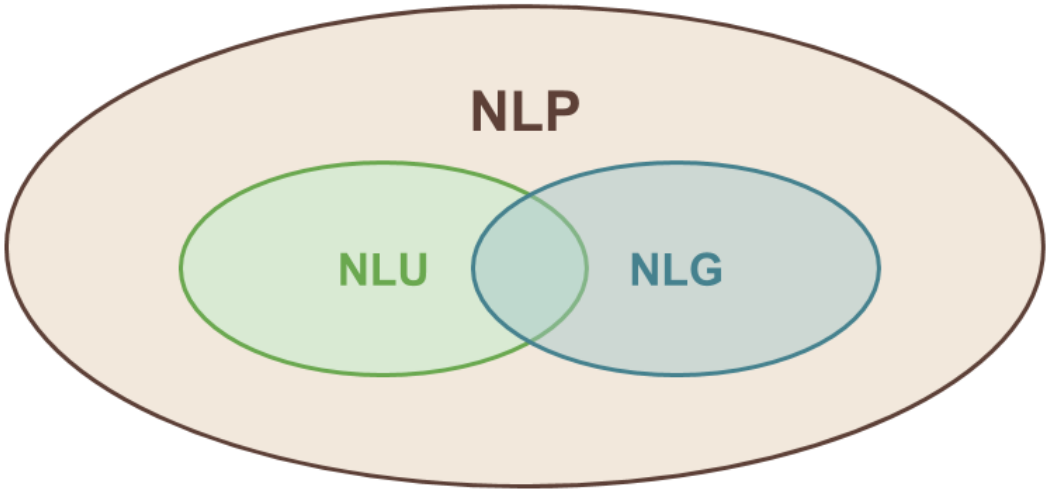

Figure 2.4. NLP - NLU - NLG.

2.2.1 Word representation

One critical point of attention in NLP is how to represent words to be understandable by the machine. For example in computer vision images are composed by pixels that are already vector of numbers. This implies that also for texts is needed a technique to translate the input text into numerical format that can be used as input of the neural network. In this section are presented the main techniques to represent words into vector representation, starting from the *one-hot vector* until the *word embedding* representation [13].

One-Hot vector representation

It is defined a vocabulary of arbitrary dimensions containing frequent words, this vocabulary can be learned by the training dataset or texts taken from internet looking for words occurring more frequently. Each word is represented by a binary vector of the same dimension of the vocabulary and in which it is placed a "1" into the component number corresponding to the word's position in the vocabulary and it is placed a "0" in all the other components of the

vector. All the words not present in the vocabulary are represented with an [unknown] token. Some examples of one-hot vector are showed in figure 2.5.

The main problem of this representation is that the inner product between any two different words is zero and then all words are seen as completely different words. With this representation is not possible for algorithms to understand relationship between words and consequently to generalize across them by understanding the context. Another weakness of this representation is that requires a long and *sparse vector* representation, of the same size of the vocabulary, for all words in the input and this impacts the number of parameters required in the neural network.

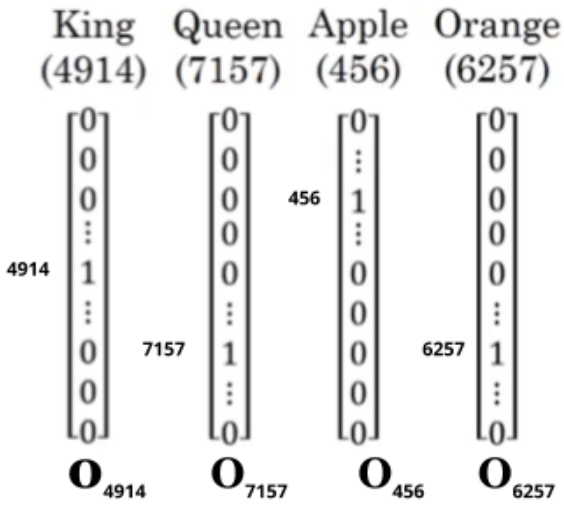

Figure 2.5. One-hot vector representation.

Word Embedding representation

Word embedding representation attempts to solve the main problems of one-hot vector representation of not understanding analogies between words and at the same time reducing the size of the input vector. To do this are *automatically learned* a set of features for each word that gives a better representation of them. This featurized representation consists of a fixed size *dense vector* of continues values for each word in the vocabulary as shows in figure 2.6. Also in this case the vocabulary can contains more frequent words learned in different ways and unfamiliar words are represented by the [unknown] token.

As explained for the hidden layers of neural networks, also each feature of word embedding is not human interpretable with components like gender, age and so on and than actuates as a black-box. The model automatically discovers the features in a way that similar words have a similar representation, in this way are learned analogies and difference between different words. The featurized representation of word embedding can captures the following relationship by taking the embedding vector corresponding to each word:

$$King : Man = Queen : Woman \tag{2.1}$$

$$E_{\text{king}} - E_{\text{man}} + E_{\text{queen}} = E_{\text{woman}} \tag{2.2}$$

Word embedding can understand that king is to man as queen is to woman for example and then learns that king and woman differentiate only for gender.

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

Features

E_{9853}

Figure 2.6. Word Embedding representation.

Visualizing Word Embedding

Using techniques like *PCA* that stands for *Principal Components Analysis* or *t-SNE* that stands for *t-distributed Stochastic Neighbor Embedding* it is possible to reduce the number of components of the featurized representation of word embedding. Reducing the embeddings to two dimensional vectors it is possible to visualize it on a 2D plot. From the plot can be noticed that related words tend to get grouped together and then word embedding learns

similar features, and then similar vectors, for concepts that should be related as showed in figure 2.7.

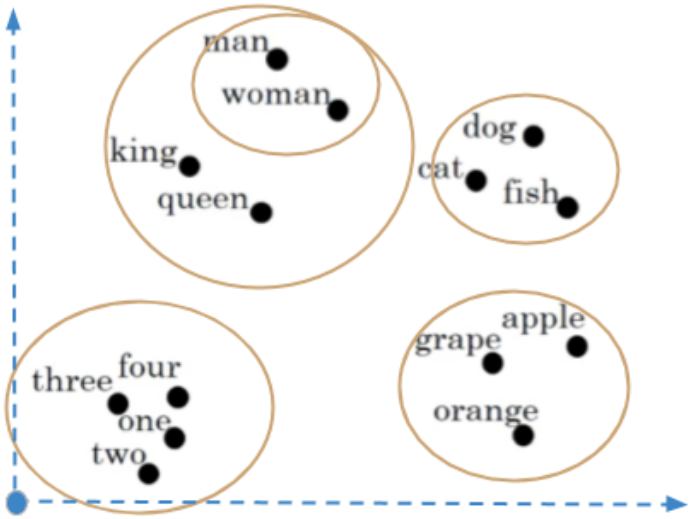

Figure 2.7. Visualizing Word Embedding on 2D plot.

Pre-trained Word Embedding models

There are different pre-trained models available open source from internet that allow to convert words into word embedding, the principals are:

- *Word2Vec*: can use two different methods to produce word embedding, both use two-layer neural networks trained to reconstruct linguistic contexts of words [11]:
  - *CBOW*: stands for Continuous Bag-Of-Words, it is the faster method and it gives a better representations of more frequent words.
  - *Skip-Gram*: is slower compared with the other method but gives a better representation of rare words.
- *GloVe*: stands for Global Vectors for word representation [16] and it is trained to understand meaning of words by observing word-word co-occurrence probabilities in order to create an encoding featurized representation for each word of the vocabulary.

The principal limitation of these types of embedding is that features for each word are static, this means that don't change based on the context to which the words are inserted to. Another limitation is that the featurized representation is learned by models that are unidirectional and then takes only the left-context (what there is before) or the right-context (what there is after) of the current word. These concepts will be clarified in the next sections.

Bias in Word Embedding

Due to the training process from large training set composed of unlabeled text taken from the internet, word embedding can reflects some gender, age, ethnicity, sexual orientation and others biases. This is due to the fact that texts are written by people and then for their nature reflecting this kind of biases. below are showed some examples of bias in word embeddings [2]:

For example if is asked to the model man is to computer programmer as woman is to what?

$$computerprogrammer : man =? : woman \tag{2.3}$$

It would be expected that word embeddings answers that man is to computer programmer as woman is to computer programmer by capturing the following relationship:

$$computerprogrammer : man = computerprogrammer : woman \tag{2.4}$$

Because computer programmer is gender neutral, but can happen that word embedding, after the training on text from internet, will capture the following relationship:

$$computerprogrammer - man + woman = homemaker \tag{2.5}$$

Word embedding has learned that man is to computer programmer as woman is to homemaker.

Another example is :

$$Doctor : He =? : She \tag{2.6}$$

It would be expected that:

$$Doctor : He = Doctor : She \tag{2.7}$$

But some research studies have shown the the model can learn:

$$Doctor - He + She = Babysitter \tag{2.8}$$

These two examples are two problems of gender bias when learning the word embedding representation. During the training process probably the input texts have a displacement where the doctor or the computer programmer was a man due to the fact that also humans have biases. This is one of the main problem of word embedding representation.

Remove Bias from Word Embedding

It is important, after the training of word embedding, to verify if the model is biased or not. There are some techniques in literature to looking for and removing bias [2]. Mainly, the idea consists of looking for a bias direction by taking the difference between words which differ only for this type of bias and make sure that words that should be neutral for this bias are at the same distance from the other ones. For example, subtracting the word embedding representation of girl and boy, grandmother and grandfather, she and he could found the gender bias direction because these words may differ only for gender. Putting in the other axis all the non-bias components of word embedding have to make sure that other words, that are gender neutral, have the same distance between couple of words that differ only for gender as showed in figure 2.8. In this example there is bias because babysitter, that should be gender neutral, is nearest to words like grandmother, girl and she that belongs to female gender, while doctor is nearest to words like grandfather, boy and he that are male gender. To remove bias need to reduce the distance of these neutral words from the non-bias direction axis to make sure that the distance from gender biases words is the same. This is an example where is taken into account one type of bias, that in this case is the gender, at a time but in reality are taken into account more types of bias together. Furthermore word embedding is not a 2D dimensional representation but an high-dimensional representation and than is very more complex process.

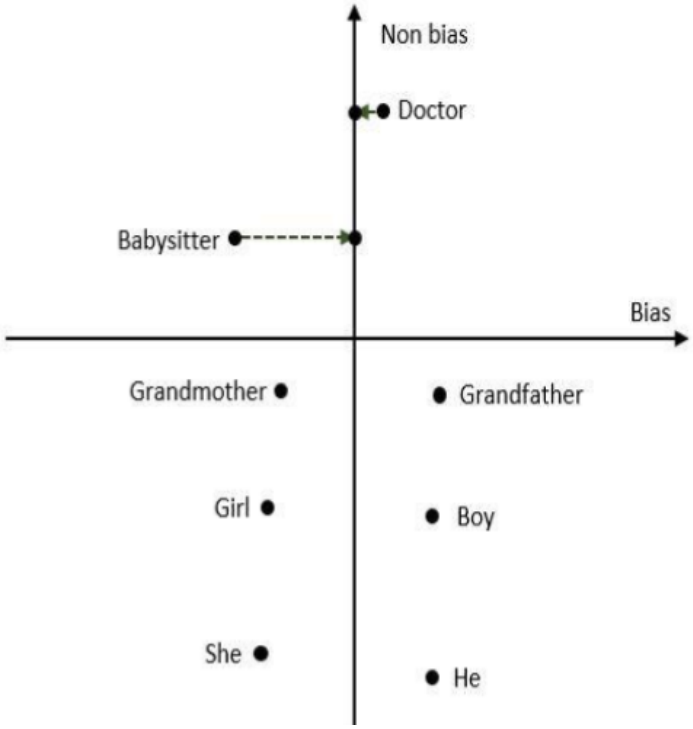

Figure 2.8. Remove bias from embedding.

2.2.2 Sequence models

Once have defined how to represent words with a machine understandable representation, and nowadays the most widespread one is the word embedding representation, now is needed to define which kind of models are more suitable to work on natural language process tasks and input texts. Standard feed forward fully-connected neural networks are not suitable of NLP tasks for different reasons:

1. Inputs and outputs can have different lengths in different input examples because input texts have not fixed lengths. One possible solution can be to assign a fixed maximum size for both input and output and padding each input and output until the max size but this solution not works very well.

2. Standard neural network architectures don't share features learned across different positions of texts. Input text is a sequential data and when computing one word need taking into account also words that are before and after in order to figure out the context of that word.

3. With one-hot vectors or word embedding input representation, the neural network will require a huge number of parameters. The number of parameters is directly proportional to the number of input words, the number of components for each word and the number of neuron in the hidden states of a standard fully-connected neural network.

RNN - Recurrent Neural Networks

RNN stands for *Recurrent Neural Network* and is a type of sequence models that scans through the network left-to-right or right-to-left. For this reason RNNs are unidirectional. At each time step, when attempts to predict Y_t, it uses information about the input at current time step but also information about the activation function from the previous time step, if works left-to-right, or the next time step if works right-to-left. The parameters and then the weights used in each time step are shared. For this reason the number of parameters is much less than would be needed by using a standard neural network.

The figure 2.9 shows an example of left-to-right RNN architecture, where W_{aa}, W_{ax} and W_{ya} are the shared weights matrices parameters.

The activation function a at each time step t is calculated as follows:

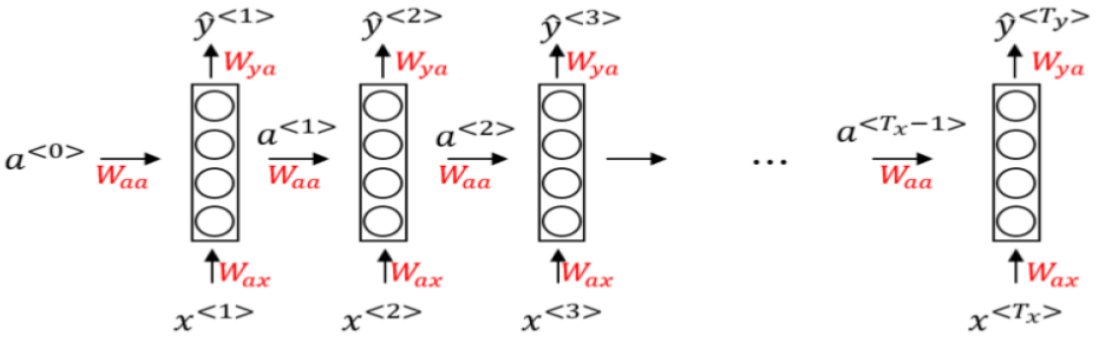

Figure 2.9. Recurrent neural network.

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{2.9}$$

Then each predicted output $\hat{y}$ at time step t is calculated as follows:

$$\hat{y} = g(W_{ya}a^{<t>} + b_y) \tag{2.10}$$

Where g can be an activation function like *Sigmoid* in case of binary classification, or *Softmax* or *Relu* otherwise. As the equation suggests the prediction at time t called $\hat{y}^{<t>}$ depends on the current input $x^{<t>}$ and from the previous activation function $a^{<t-1>}$.

RNN architectures

There are different possible RNN architectures depending on the task and then on the number of inputs and outputs. For example the *Many to one* architecture is suitable for sentiment classification. The *Many to many* architecture with $T_x = T_y$ is suitable for named entity recognition. Different types of RNN architectures are illustrated in figure 2.10.

RNN: Encoder - Decoder Architecture

One of the most interesting architecture is the *Encoder - Decoder architecture*. It is a *Many to many* architecture where the number of inputs can be

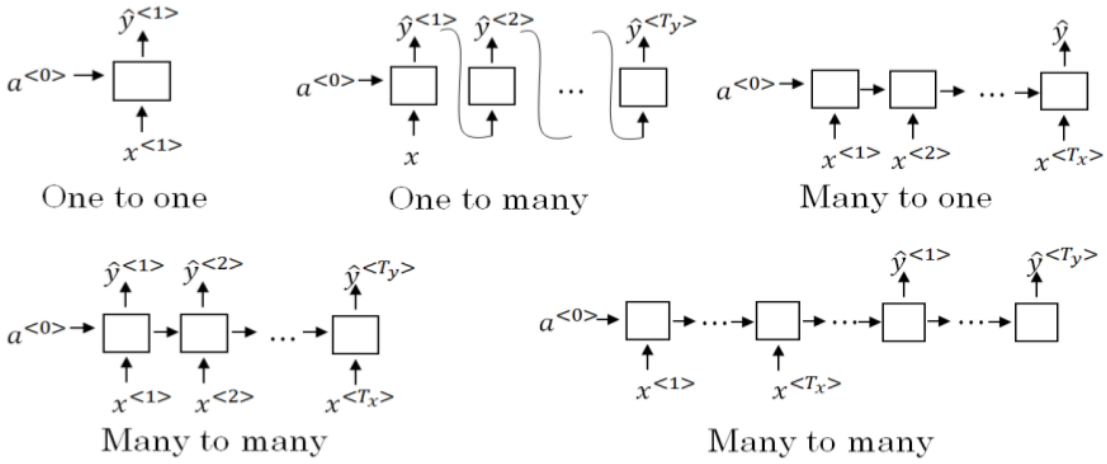

Figure 2.10. RNN: Different architectures.

different from the number of outputs, then T_x != T_y. This architecture is very suitable for tasks like machine text translation where is not possible to assume that the number of inputs is equal to the number of outputs.

The encoder maps the input $X = (x_1,...x_n)$ from a sequence of symbols representation to a continue one $Z = (z_1,...,z_n)$. This continuous representation of the input is feeded in the decoder where it generates an output sequence of symbols $Y = (y_1,...,y_m)$ one element at a time by consuming the previous generated symbol. In other word the encoder takes the input and transforms it into a *fixed size vectorized representation of the whole sequence*, this vectorized representation is used as the input of the decoder part. The structure of the encoder decoder architecture is showed in figure 2.11.

Problems of standard RNNs

Standard recurrent neural networks have three main problems:

1. *Unidirectional*: standard RNNs take only left-to-right or right-to-left context and then only what there is before or after the word. This is a very big limitation because the context of words needs to be figured out by looking both what there is before and after the word at the same time in order to have a better representation.

2. *Vanishing Gradient*: RNNs suffer of vanishing gradient problem and the

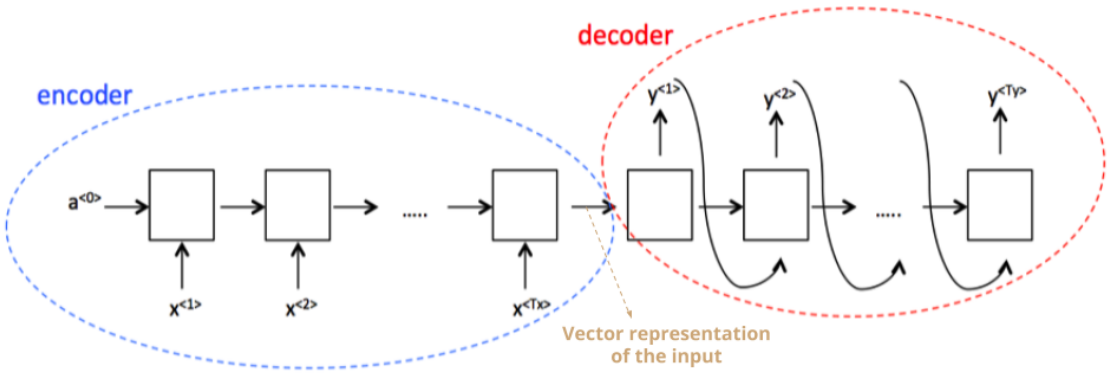

Figure 2.11. RNN: Encoder - Decoder architecture.

fixed-length vector representation of the input source made by the encoder is the bottleneck in that the encoder needs to compress all the necessary information into a fixed size representation. When the size of the input source increases then the compression is higher and consequently the fixed representation is worse.

3. *Not parallelizable*: parallelization is not possible due to the sequentiality of the model and this implies lower speed and performance.

BRNN - Bidirectional Recurrent Neural Networks

BRNN stands for *Bidirectional Recurrent Neural Networks* and, as the name suggests, try to solve the unidirectional problem of standard RNNs. With BRNN, when predicting the output Y_t, it attempts to take information from both what is in the "past" and what is in the "future". To do this BRNN combines a left-to-right and a right-to-left component and then, when predicting output at Y_t, takes both the backward state and the forward state. BRNN reduces the problem of unidirectional context of standard RNN but not solves it completely. A bidirectional recurrent neural network can be represented as the figure 2.12.

In this case are calculated two activation functions, one taking the left-context and one taking the right-context:

$$\overleftarrow{a}^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{2.11}$$

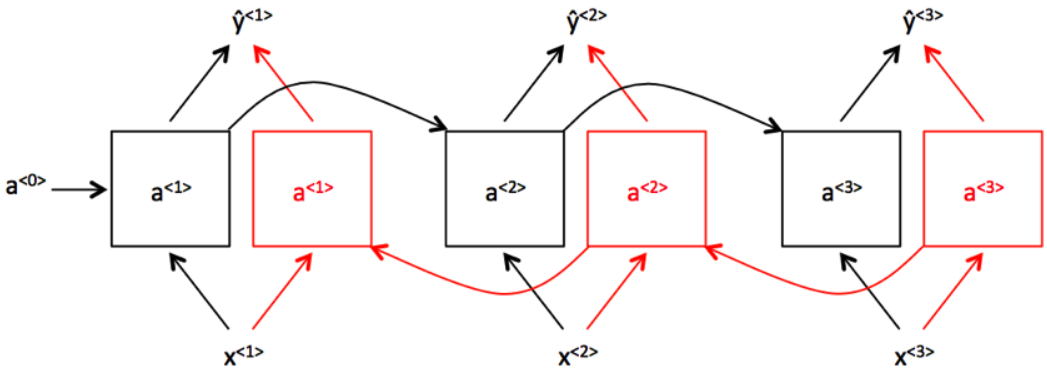

Figure 2.12. BRNN: Bidirectional Recurrent Neural Network architecture.

$$\overrightarrow{a}^{<t>} = g(W_{aa}a^{<t+1>} + W_{ax}x^{<t>} + b_a) \tag{2.12}$$

This implies that the activation at time t is:

$$a^{<t>} = [\overleftarrow{a}^{<t>}, \overrightarrow{a}^{<t>}] \tag{2.13}$$

And the final output prediction $\hat{y}^{<t>}$ at time t is calculated as:

$$\hat{y} = g(W_{ya}a^{<t>} + b_y) \tag{2.14}$$

Where substituting $a^{<t>}$ with the equation 2.13:

$$\hat{y} = g(W_{ya}[\overleftarrow{a}^{<t>}, \overrightarrow{a}^{<t>}] + b_y) \tag{2.15}$$

The last equation shows that, when attempts to predict $\hat{y}^{<t>}$, the BRNN takes information from both past and future by multiplying the two activation functions for the corresponding weights.

Solving vanishing gradient problems

To solve the problem of keep a good representation of very long sentences have been implemented different methods:

- *GRU* and *LSTM*: one possible way is to substitute the standard rnn unit with more complex units like *GRU* [3] that stands for *Gated Recurrent Unit* and *LSTM* [6] that stands for *Long Short Term Memory* that make use of concepts like memory cell and gates to keep some information

about the input sentence. However details about GRU and LSTM are out of the scope of this paper and for more information are reported the references.

- *Attention*: using attention mechanisms in order to figure out the context of each word. More details are provided in the following sections.

2.2.3 Attention mechanisms

Attention mechanisms [1] attempt to solve the problem of encoder - decoder RNNs and BRNNs of memorizing the whole representation of very long sentences with a vectorized fixed-length representation as output of the encoder block. The intuition below the attention mechanisms is that when predicting the output, for each word, try to figure out the context of this word and assign weights that tells how much "attention" this current output have to pay to each word of the input sentence.

With standard encoder - decoder RNN architectures the encoder produces a single context vector c and the decoder predicts the next word y_t given the context c and the previously predicted words $\{y_1,...,y_{t-1}\}$. Note that the context vector c is the same for all the predicted words y_t. With attention, instead of memorizing the whole sentence and then predicting the output, it looks part of the sentence at a time.

An example of attention based architecture is showed in figure 2.13. The encoder block is composed by a bidirectional recurrent neural network in order to take both left and right context of the current word.

The hidden state h_j is composed by the concatenation of $\overleftarrow{h}$ and $\overrightarrow{h}$:

$$h_j = [\overleftarrow{h_j}, \overrightarrow{h_j}] \tag{2.16}$$

Then the context vector c_i for the word *i-th* is calculated as follows:

$$c_i = \sum_{j=1}^{Tx} \alpha_{ij} h_j \tag{2.17}$$

Where the weights α_{ij} of each annotation h_j is computed as:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{2.18}$$

where:

$$e_{ij} = a(S_{i-1}, h_j) \tag{2.19}$$

The calculated e is defined as the *energy* and is the importance of h_j with respect of the previous hidden state of the decoder S_{i-1} to predict y_i and the state S_i. In other words the decoder can decide to which parts of the input sentence have to pay *attention*. This type of attention is called *Additive Attention*.

The attention weights α_{ij} are learned and calculated by feeding in a feed-forward neural network S_{t-1} and h_j.

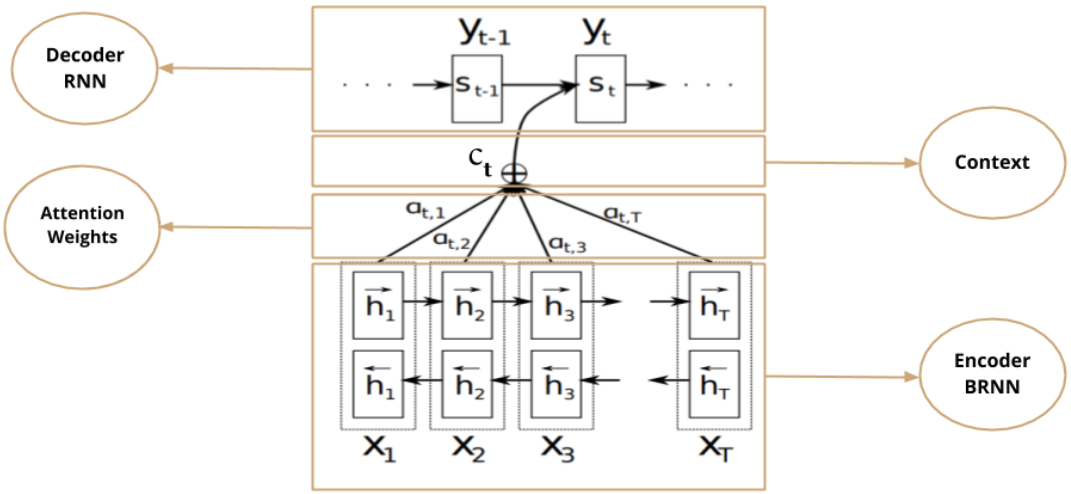

Figure 2.13. Attention blocks.

2.2.4 The Transformer architecture

The *Transformer* [23] was introduced with the Google's paper *"Attention is all you need"* and has drastically changed the state-of-the-art for sequence models [22]. It attempts to solve the main problems of standard RNNs and BRNNs like *unidirectional context*, *parallelization* and *single fixed vector representation of input* together.

The Transformer follows the overall encoder-decoder architecture but without using any kind of recurrence or convolution, in order to allow the parallelization, but using instead *stacked self-attention* and *point-wise, fully connected layers* for both encoder and decoder. The overall transformer architecture is showed in figure 2.14.

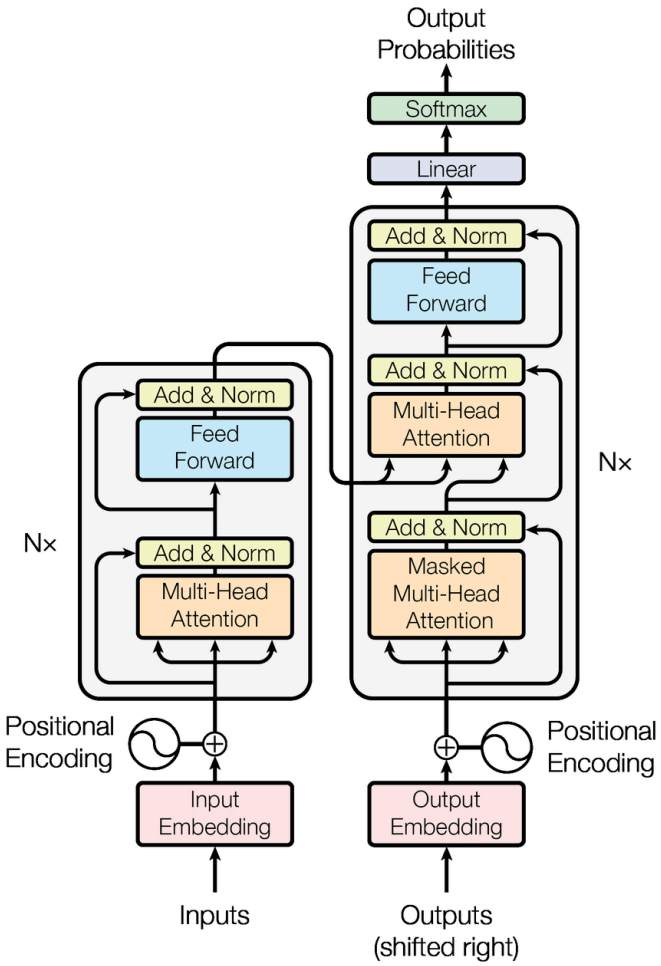

Figure 2.14. The Transformer - model architecture.

Encoder

The encoder [19] is composed by N identical layers stacked together and, in the original paper, the number of layers N is equal to 6. Furthermore, each layer is composed by two sub-layers:

1. The first sub-layer is a *multi-head self-attention mechanism*.
2. The second sub-layer is a *position-wise fully connected feed-forward network*.

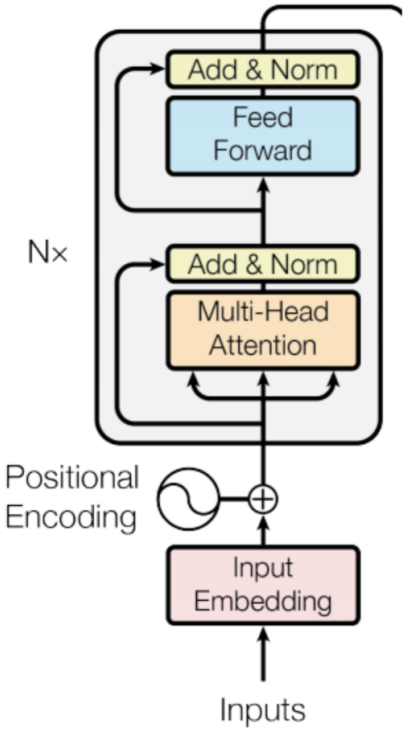

Figure 2.15. The Transformer - Encoder.

Decoder

The decoder [18], as the encoder, is composed by N identical layers stacked together and also in this case the number of layers N is equal to 6. In this case each layer is composed by three sub-layers:

1. The first sub-layer is a *masked multi-head attention mechanism*.
2. The second sub-layer is a *multi-head self-attention mechanism*.
3. The third sub-layer is a *position-wise fully connected feed-forward network*.

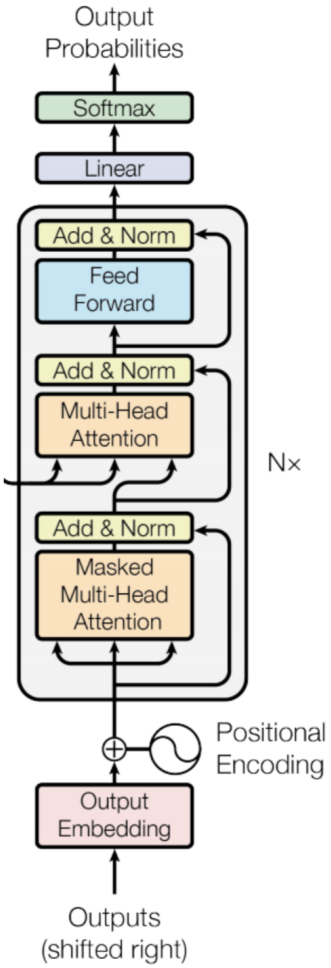

Figure 2.16. The Transformer - Decoder.

Transformer attention

An attention function can be defined as a mapping between a *query* and a set of *key-value* pairs to an *output*. The *query*, *keys*, *values*, and *output* are all vectors. The output is computed as a weighted sum of the values and the weight assigned to each value is computed by using a compatibility function of the query with the corresponding key. The particular type of attention implemented in the transformer is called *"Scaled Dot-Product Attention"*.

Scaled dot-product attention

Instead of using the additive attention explained the section 2.2.3, the Transformer uses the *multiplicative attention* because, using optimized matrix multiplication code, results much faster and space-efficient.

Given the following input:

- Q: Queries of dimension d_k
- K: Keys of dimension d_k
- V: Values of dimension d_v

The attention is computed as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.20}$$

In practice the attention function is computed on a set of queries at the same time by packing them together into a matrix Q, also keys and values are packed together into matrices K and V. The values V are weighted by a softmax function applied on the dot product of queries Q with keys K, scaled by a factor of $\sqrt{d_k}$.

Multi-head attention

Instead of using a single attention, the transformer uses h different attention calculations with different weight matrices. These different computations are computed in parallel and the result of each of it is called *head* and corresponds to a *scaled dot-product attention* explained before. This particular kind of attention is called *Multi-head attention* and the final result is computed as the *concatenation* of all heads as showed in figure 2.18.

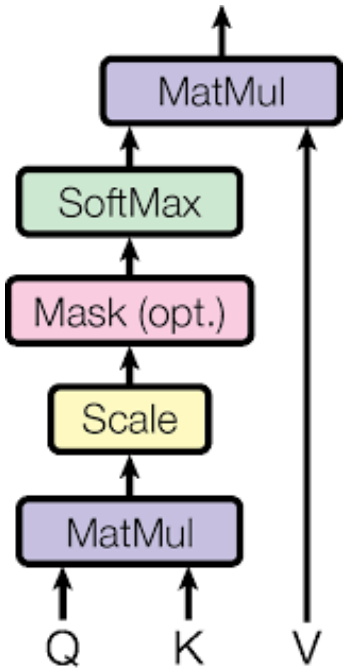

Figure 2.17. The Transformer - scaled dot-product attention.

In mathematical terms:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2.21}$$

where:

$$head_i = Attention(QW_i^Q, KW_i^K, W_i^Q, VW_i^V) \tag{2.22}$$

Using different attention head, allows the model to capture different aspects of the input and then the model can takes information from different parts of the input text jointly.

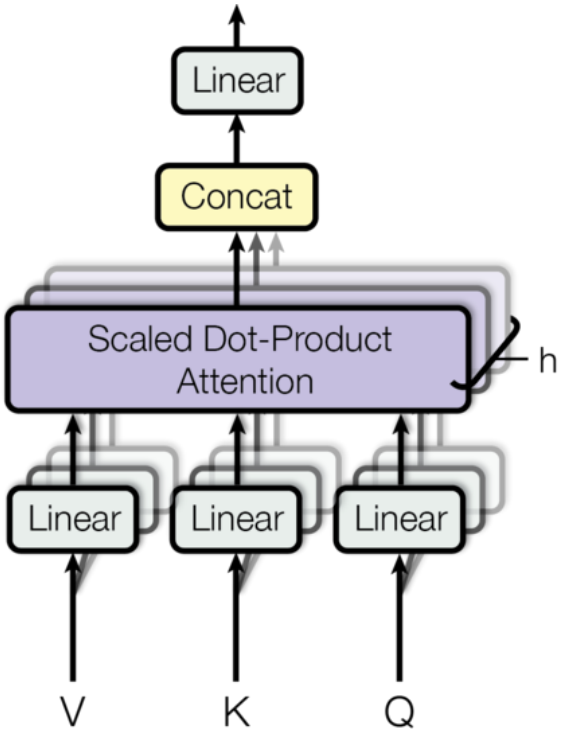

Figure 2.18. The Transformer - multi-head attention.

Feed-forward network

Each encoder and each decoder contains a *fully connected feed-forward network*. This network applies two linear transformations and a Relu activation in between to each position separately and identically.

The two linear transformations are the same across different positions but they have, in each layer, different parameters. The dimensionality is $d_{model} = 512$ for input and output and $d_{ff} = 2048$ for the inner layer.

Embedding ans Softmax

The Transformer makes use of a learned *embeddings* to convert the input tokens and output tokens to vector of d_{model} dimension. It makes use of a learned *linear transformation* and a *softmax* function in order to convert decoder output to predicted next-token probabilities. The weights between the two embedding layers and the pre-softmax are shared. The positional encoding of the embedding is added at the beginning of the encoder and decoder stacks.

Positional encoding

The model not uses both recurrence and convolution, this implies that is needed a way to represent the position of each token into the sequence. To do this it is added a *positional embedding* to the input embeddings at the beginning of both encoder and decoder stacks. The positional embedding has the same dimension of the embedding and is equal to d_{model} and then can be added to the normal token embedding.

2.3 BERT - Bidirectional Encoder Representation from Transformers

Figure 2.19. Google BERT.

BERT [4] stands for *Bidirectional Encoder Representation from Transformers* and is a new language model representation that has changed drastically the state-of-the-art in natural language processing and was presented with a Google's paper in the 2018. As the name suggests the key features of BERT are the *Bidirectionality* of the model obtained with the use of the *Encoder* part of the *Transformer* architecture 2.2.4. As explained in the previous section the bidirectionality is obtained without the use of any type of recurrence or convolution but exploiting a revolutionary type of pre-training. The name suggests also that BERT uses only the *Encoder* part of the transformer to provide a *Representation* of the words. For these reasons the main objective of BERT is to create a language representation model that can be used for different tasks or to exploit contextualized word embeddings features by stacking different transformer encoder layers one over the other.

2.3.1 Input representation

Wordpieces representation

Instead of using one token for each frequent word and the [unknown] token for out-of-vocabulary words like standards static word embedding, it uses a particular type of word embedding representation called *Wordpiece Embeddings*. Frequent words, present in the vocabulary, are treated as normal tokens, out-of-vocabulary words instead are divided in subwords present in the vocabulary and are called wordpieces. For notation, each wordpiece is preceded by the prefix ##. Also each single character is inserted in the vocabulary, in this way it is possible to represent any kind of out-of-vocabulary word into wordpieces. Using this notation for the input, the [unknown] tokens is not necessary.

The advantages of wordpiece tokenization are:

- It reduces the size of the vocabulary.
- It increases the amount of data available to represent features for each word and then gives a better representation of out-of-vocabulary words.

Example: *This is an example of wordpiece embeddings*

[“this”,“is”,“an”,“example”,“of”,“word”,“##piece”,“em”,“##bed”,“##ding”,“##s”]

For example the word *"embeddings"*, if is not founded in the vocabulary, it is splitted into subwords called wordpieces that give an approximated representation of the word. One possibility is to average the values of each wordpiece in order to obtain a single featurized representation of the whole word.

Sentences representation

The initial token of every sequence is a special classification token called [CLS] that is used as aggregate representation of the whole sentence and is useful in case of classification tasks. Sentence pair are packed together into a single sentence and are separated with another special token called [SEP].

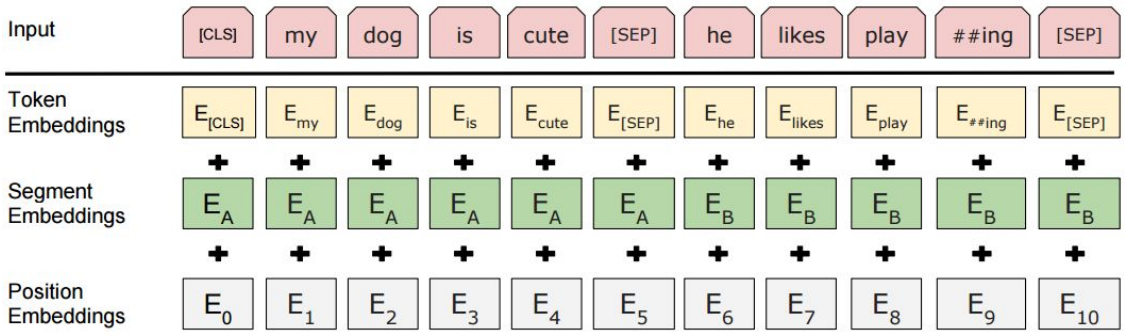

Figure 2.20. BERT - input representation.

Contextualized embeddings

As explained before, BERT's architecture is composed by many transformer encoder layers, the transformer not has recurrence and then not read one input word at a time but the whole input is read at once. Due to this it needs a way to represent the position of the word in the input; for this reason is added the positional embedding. The final embedding for each wordpiece is calculated as the sum of *Token Embedding*, *Segment Embedding* and *Position Embedding* as showed in figure 2.20.

2.3.2 Pre-Training

BERT uses two semi-supervised tasks for pre-training:

- MLM - Masked Language Model
- NSP - Next Sentence Prediction

The model was trained for this two tasks simultaneously with a dataset built from BookCorpus (800M words) and English Wikipedia (2,500M words). The goal of pre-training [7] is minimizing the combined loss function of these two tasks. The combination of these two tasks allows BERT to inherit bidirectionality, to be precise BERT can be defined non-directional because avoid using unidirectional left-to-right, right-to-left or the combination of the them like traditional RNN or BRNN language models.

MLM - Masked Language Model

The main objective of this task is to learn a featurized representation for each wordpiece in the vocabulary. To do this are evaluated the 15% of all

wordpieces of the input text. For each of this wordpieces selected randomly:

- With a percentage of 80% the wordpiece is replaced with the [MASK] token.
- With a percentage of 10% it is replaced with another random token.
- With a percentage of 10% the wordpiece remain unchanged.

The 15% of wordpieces selected are not all replaced with the [MASk] token to reduce the mismatch between pre-training and fine-tuning because this [MASK] token does not appear in the fine-tuning phase.

The model attempts to predict the original values of the masked words by feeding the final hidden vector of each [MASK] token into an output softmax over the vocabulary, as showed in figure 2.21. With this prediction the model learns a featurized representation of each wordpiece (word embedding) that is bidirectional because the input text is readed in one shot and then for each predicted masked token there are both left and right contexts.

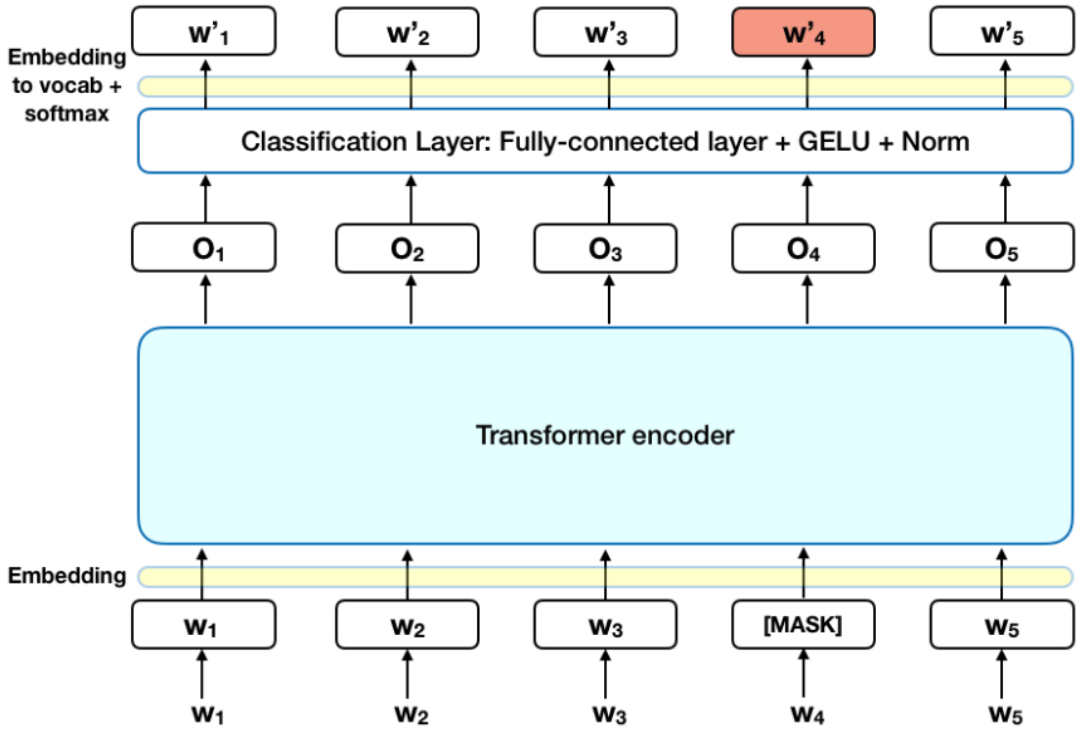

Figure 2.21. BERT - Pre-training: Masked Language Model.

NSP - Next Sentence Prediction

The main objective of this task is to learn relationship between two sentences. This aspect is fundamental for tasks like Question Answering (QA) or Natural Language Inference (NLI) and it is not directly captured by MLM task. To do this, the model is pre-trained for a binarized next sentence prediction task and for each input pre-training example are chosen sentences A and sentences B where:

- The 50% of time the sentence B is the actual next sentence that follows A and the label is *isNext*
- The 50% of time the sentence B is a random sentence from the corpus and the label is *notNext*

The probability that the next sentence is the actual next sentence or a random sentence is predicted with a classification layer and a softmax.

Pre-trained models

There are a set of pre-trained models of BERT available:

- *BERT-Base, Uncased*: 12-layer, 768-hidden, 12-heads, 110M parameters
- *BERT-Large, Uncased*: 24-layer, 1024-hidden, 16-heads, 340M parameters
- *BERT-Based, Cased*: 12-layer, 768-hidden, 12-heads, 110M parameters
- *BERT-Large, Cased*: 24-layer, 1024-hidden, 16-heads, 340M parameters
- *BERT-Base, Multilingual Cased*: 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- *BERT-Base, Multilingual Uncased*: 102 languages,12-layer, 768-hidden, 12-heads, 110M parameters

These models differ by size (Base - Large), language (English - Multilingual) and case sensitivity (Cased - Uncased).
Exploiting transfer learning, with the fine-tuning phase, it is possible starting from one of these pre-trained version, in relative small amount of time, training a model for a specific task also with much less training data.

2.3.3 Fine-Tuning

Exploiting the potential of *transfer learning*, starting from one of the pre-trained models, the fine-tuning allows to specialize the model in the required task. BERT can be fine-tuned for many specific tasks like sentiment analysis, question answering, sentence classification, named entity recognition etc. Fine-tuning can be done, for example, by adding a classification layer and with the specific labeled dataset fine-tuning all parameter end-to-end. For the fine-tuning phase most of hyperparameters remain the same of the pre-training; Moreover this phase is relative inexpensive compared with pre-training.

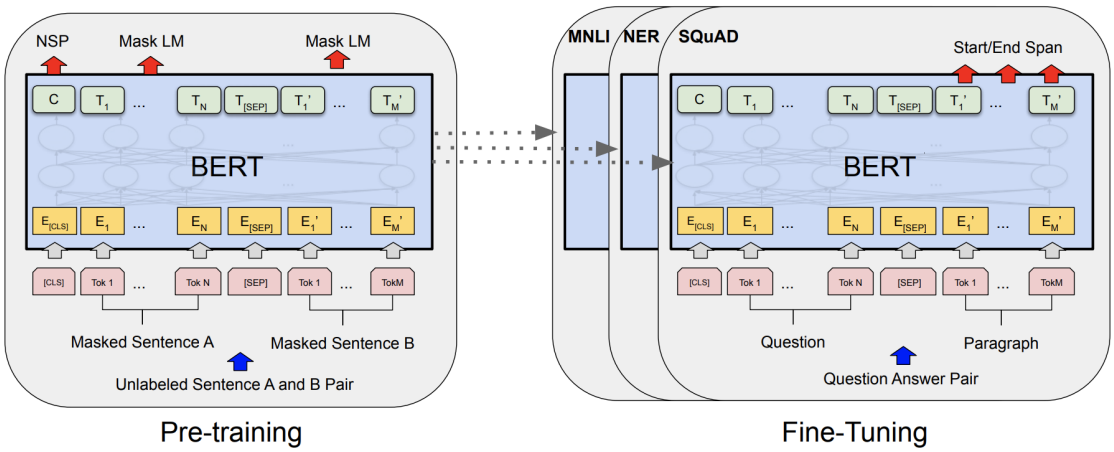

Figure 2.22. BERT - fine tuning.

2.3.4 Contextualized features extraction

Fine-tuning BERT can create a model for a specialized task. But beyond that, BERT can be used also to extract and create contextualized word embedding. The word embedding extracted can be used in a totally different network, due to the fact that some tasks are not easily represented by a transformer architecture and starting from the BERT embedding can be applied transfer learning. Using this transfer learning can have lot of computational benefits because pre-training a model is a very expensive task.

The output of each layer of the model can be used as possible featurized representation. The model architecture is composed by a stack of transformer

encoder blocks, each one takes, for each word, an input of the embedding dimension and creates an output of the same dimension. In this way the output of each encoder can be used as the input of the next encoder and each one can provide a possible featurized representation of the word as showed in figure 2.23.

One question can be: which layer should be used in order to have the better representation of words?

The figure 2.23 shows an example of Dev F1 score by using BERT contextualized embedding for a named entity recognition task. From the figure 2.24 emerges that the *concatenation* of the *sum* of the last four encoder layers gives a very accurate representation of embedding in the case of *Named Entity Recognition* but these scores are similar also for other tasks.

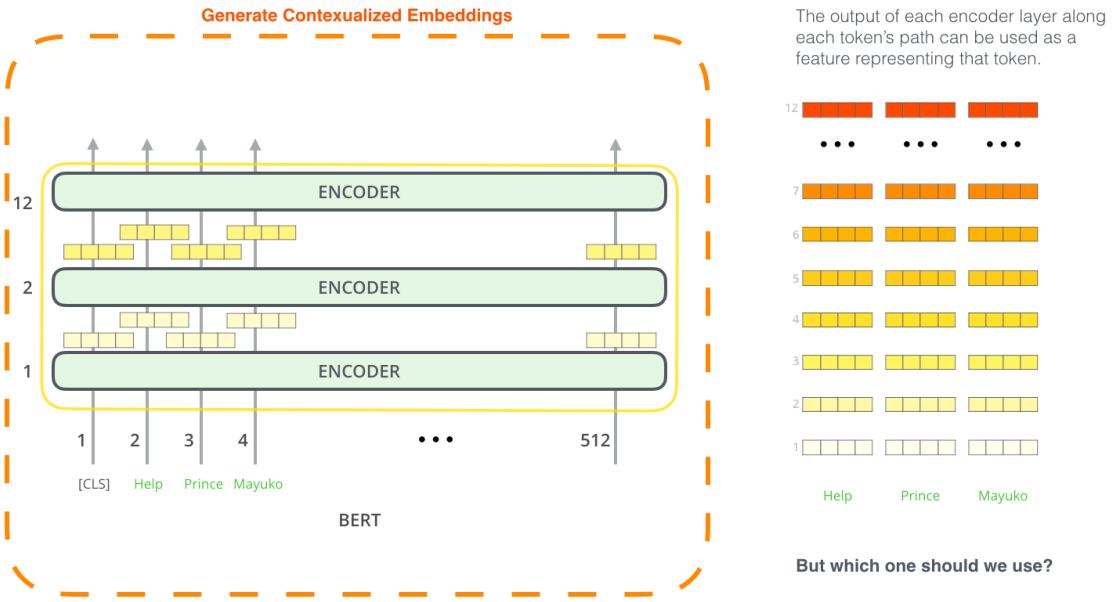

Figure 2.23. BERT - Contextualized Embeddings.

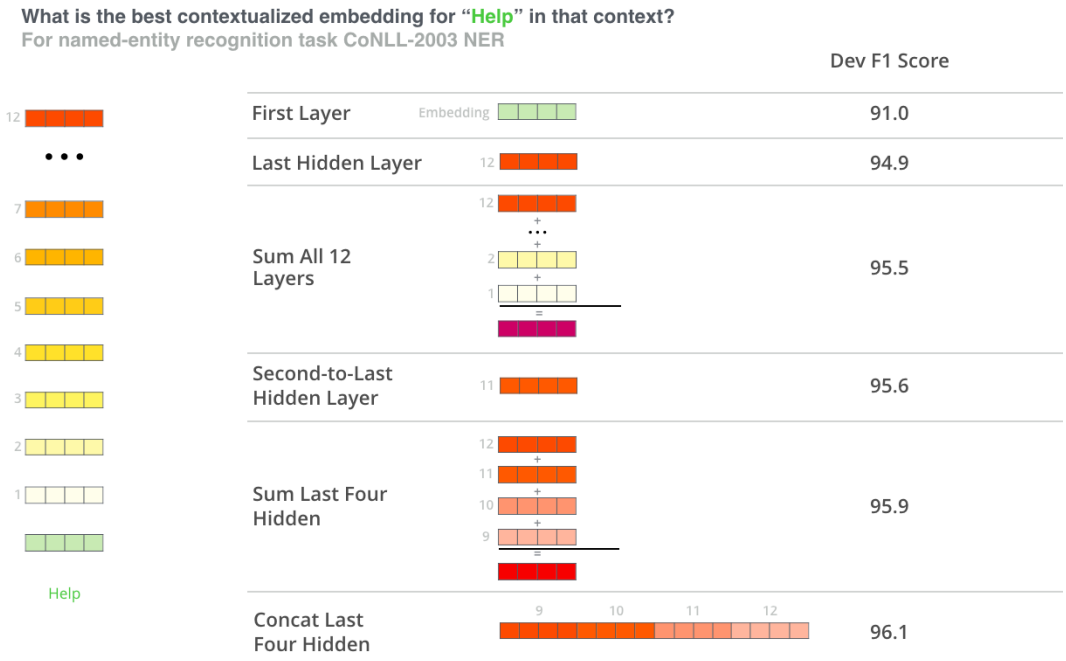

Figure 2.24. BERT - Contextualized Embedding: Dev F1 Score on NER.

2.3.5 Visualizing attention

Another key aspect of BERT is that makes use only of attention mechanisms to draw dependencies between input and output. There was lot of studies on this aspect and are available papers [25] and tools that help to see the attention for each word in each layer with respect of each input word as showed in figure 2.25.

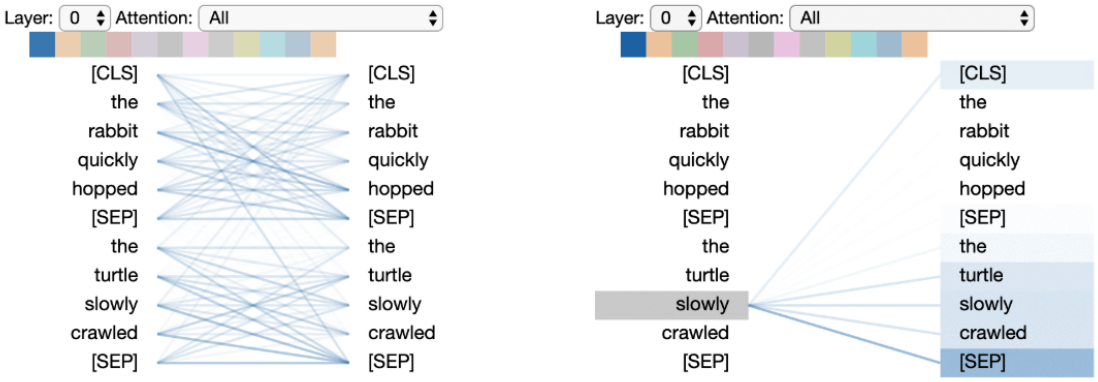

Figure 2.25. BERT - Visualizing attention.

2.4 Explainability and xAI

Nowadays more and more decisions, often critical, are taken by decision-making algorithms that are complex machine learning models with millions of parameters that by their nature are intrinsically black-box. These models are trained with enormous amount of data that people produce in every day life like web searches, purchases, social networks, movements and so on. these decisions affect people's life for example for discarding cv in job applications, for loan bank grant, autonomous driving cars and for a lot of other aspects. In some cases these algorithms could also exceed the human level of accuracy and performance making more accurate and precision predictions then those of humans.

The question that arises is: Why is needed an *explanation*?

One possible reason is that data may inherit human *biases*, *racism* and *prejudices* that can cause *unfair* and *wrong* decisions. For example the job application could discard some curriculum for a software developer position only because the applicant was female, or maybe a loan grant algorithm could reject a loan request only because is done by a person belonging to an ethnic minority. In this case exaplaining the model can figure out why it reflects *bias*.

Another possible reason is that companies that want use these models to automate some processes have to *trust* these algorithms. The *trustability* is a key concept for machine learning's adoption and acceptance because people and companies will not use models that don't trust. Explain a model can show if it takes good or bad decisions and then decide if use it or not.

Moreover an explanation can be required for *legislative* and *regulatory* reasons. *GDPR* that stands for General Data Protection Regulation has impacted the AI world mainly with the following articles:

- *Article 5*: in summary it says that is required for organizations to minimize the amount of collected data, furthermore companies may process data in a *transparent* manner in relation to the data subject.
- *Article 22*: in summary it says that, when companies use AI to take important decisions about peoples, then the data subject has the right to have *human review* that decision.

Resuming the example of the loan bank grant, with these regulations, the subject corresponds to the human that requires the loan and has the right to understand in human terms and in a transparent manner the reason of the decision. In addition, also the bank need to be sure and trust the decision taken by the algorithm for not giving a loan to wrong people.

Another possible reason is the *safety*. Some decision models work in critical contexts where a wrong prediction can cause the death of a person. For example if the autonomous driving car makes a wrong decision puts the life of the driver and pedestrians at risk. Another example of critical decision is the cancer detection algorithm. Also in this case a wrong prediction can put a person's life at risk, moreover the patient will want a human understandable description of the reasons he has or does not have cancer. Most of the application working in the medical field can be considered critical.

In summary, explanation is mainly needed for many different reasons like *trustability*, *fairness*, *safety*, *verifiability* and *regulatory* but there are also a lot of other reasons. In this context, it is increasingly emerging the research field of *xAI* that stands for *eXplainable Artificial Intelligence* [20].

2.4.1 Real facts examples

In this chapter are reported a set of concrete examples that can give a better idea of how these algorithms, when make wrong prediction or when present bias, can impact people's life.

Examples of bias, racism and prejudice

- *Google image-labeling*: The image classification algorithm of google classified image of black peoples as gorillas. The algorithm had an ethnic bias probably because it was trained with more images with white people then black ones.

- *Amazon AI recruiting tool*: the recruiting tool had bias against woman and then preferred male candidates on job applications with respect of female ones. It could have happened because the higher percentage of worker inside amazon was male and then the algorithm understand that male worker are "better" for this type of job or maybe also if during the training was taken texts from internet with this bias.

- *COMPAS risk of crime recidivism*: the algorithm predicts the tendency of a convicted criminal to reoffend. Black peoples were labeled almost twice as higher risk but they actually not re-offend. Also in this case the algorithm had a ethnic bias and then was racism probably for a mismatch in the training dataset.

Examples of fatal wrong prediction

- *Self-Driving Uber Car*: In 2018 an Uber self-driving test vehicle killed a woman probably because failed to recognize her as a pedestrian.

2.4.2 xAI - eXplainable Artificial Intelligence

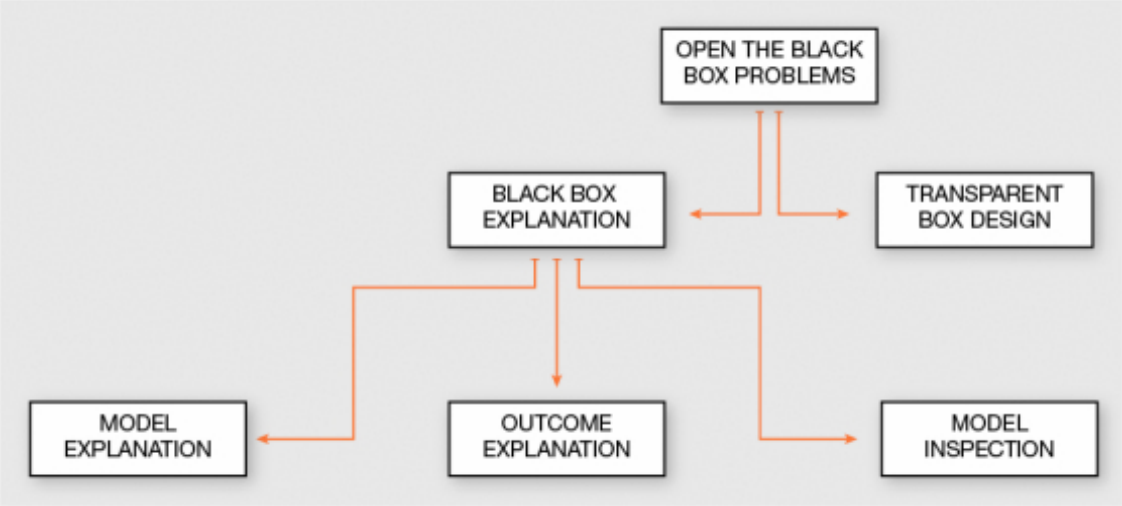

Figure 2.26. Explainability diagram

So far it has been explained why is important to make black-box models interpretable. This parts instead attempts to explain how to make black-box models more interpretable [5].

Interpretability can be defined as the ability to explain or provide the meaning in understandable terms to human.

A black-box model to be interpretable requires an *explanation*. Explanations can be divided in two macro category:

- *Global explanation*: consists of providing an explanation that allows to understand the whole logic of the model and all possible outcomes of that model. This implies that the whole model is completely interpretable.
- *Local explanation*: consists of providing an explanation that allows to understand only the reason for a specific decision or for a specific outcome of the model. This implies that the single prediction is interpretable.

Given a problem that can be solved by a black-box neural network, how can also get the transparency property?

1. *Transparent box design*: consists of directly providing a model that is locally of globally interpretable instead of using a neural network or other intrinsically black-box model. Unfortunately it is often not possible because transparent models are not as powerful as neural networks. Transparent models are for their nature easily understandable in humans terms, some examples of them are:
    - *Decision trees*: are tree-like models used for classification and prediction where exploiting the tree graph give a visual explanation of the prediction. The decision tree is composed by a set of nodes that represent the condition and the arrows that represent the outcome of the condition, until the leaf nodes that represent the class labels. Looking the path from the root of the tree passing over a set of condition and output nodes it is possible to understand the reason of the predicted class label that correspond to the leaf node of the path. Decision trees are probably the most understandable models for humans.
    - *Decision rules*: are a set of *if then* rules that given an input predict the output. Each if then statement is composed of a condition and a prediction, by combining multiple statement can create more complex prediction. This kind of models are very easily understandable by humans.
2. *Black-box explanation*: given a neural network that works well for a specific task, attempts to explain how this black-box models makes decisions. Explanation can be divided in three different types:
    - *Model explanation*: provides a *global explanation* of the black-box model by using another model *transparent by design* and then interpretable and transparent in humans terms, this model should mimic

the behaviour of the black-box model that attempts to explain. In this way the neural network it is used to takes decisions as it is more accurate and the transparent model to approximate the decision to in a human understandable way.

- *Outcome explanation*: provides a *local explanation* of the black-box model by explaining the output on a given instance, in this case it is needed to explain the reason for the specific prediction but not to understand the whole underlying logic of the model.
- *Model inspection*: consists of providing a visual or textual representation for understanding and explaining some specific property of prediction of the given black-box model. Also in this case a *local explanation* is provided.

One of the problem of explanations is that require a trade-off between performance and execution time in order to obtain interpretability.

2.4.3 Explanation techniques at the state-of-the-art

One of the most widespread technique to explain black-box model is *LIME* [17] that stands for *Local Interpretable Model-Agnostic Explanations*. The main purpose of LIME is to explain the prediction of any classifier and then it is used in very different contexts like image or text processing. It provides an explanation by approximating the black-box model with an interpretable one locally.

Apart of this, in the same paper, it is presented also another method called *SP-LIME* that stands for *Submodular Pick - Local Interpretable Model-Agnostic Explanations* that attempts to provide a global understanding of the whole model by analyzing a set of instances individually. The main strength of these two methods is that they are model agnostic, and then they are a general approach that can be applied to any kind of black-box classifier.

Another well known technique to explain black-box models is *Grad-CAM* [21] that stands for *Gradient-weighted Class Activation Mapping*. It is based on the Gradient-based Localization technique in order to provide a visual explanation of deep convolutional neural networks. This technique is strictly related to the image classification task.

Chapter 3

Proposed Methodology

In this chapter, firstly will be presented as related work the *general methodology* and then will be introduced *EBANO Text* that is the specialization of the *general methodology* in the context of natural language processing. This is due to the fact that *EBANO Text* is part of a bigger framework, where the starting project had the objective of providing explanation in computer vision field and is already implemented. If the methodology in this new context will works, then the process of explanation can becomes more task and model agnostic. The main focus of this chapter is to presented the *EBANO Text architecture* that is the core of this thesis and that shows the full process in order to obtain a *local explanation* starting from an input text and a fine-tuned BERT model. Moreover are provided lot of details on the multi-layer feature extraction method that is the more complex, interesting and innovative feature extraction method proposed.

3.1 Related work

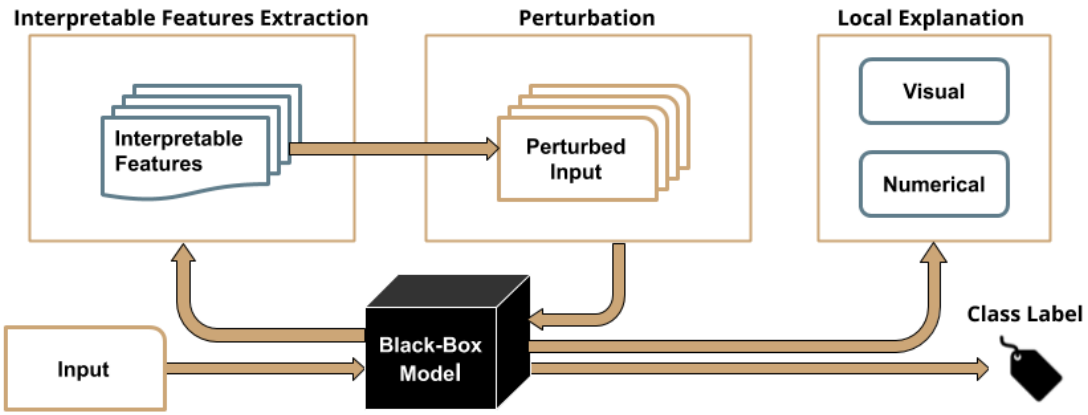

Figure 3.1. General Methodology

In this section will be explained the *general methodology* in order to provide an *outcome explanation* of a given black-box model. As explained in the previous sections an outcome explanation not gives the understanding of the whole logic of the model but explains, given an input, the reason for the specific outcome of the model and then it is provided a *Local Explanation*.

The underlying idea is to provide the explanation by applying a set of *Perturbations* over a set of *Interpretable Features* extracted from the input and then produce a *Visual* and *Numerical* explanation of the impact or *Influence* that the features have on the original output prediction of the model given that specific input. In the figure 3.1 are represented all steps of the *General Methodology*.

The proposed methodology is called *EBANO Text* and it is a part of a bigger framework called *EBANO* [24] where the common goal is the explanation of black-box models. The most consolidated part is about image processing and the explanation of some *Convolutional Neural Network*. *EBANO Text* instead has as its main purpose the explainability in the context of Natural Language Processing. What changes is the type of input that in this case is text and then the techniques for features extraction, for apply perturbations and to show the results.

3.1.1 Input and Black-Box model

In the general approach the type of *input* and the type of *black-box model* are strictly bounded. Input data is typically *unstructured data*, then the black-box model depends on it but can works for all types of deep neural networks. In the case of EBANO on images the black-box models explained are *Convolutional Neural Networks* as vgg_16, vgg_19, inception_res_net_v3 and so on.

As explained in section 2.2.2 this type of neural networks are not suitable for natural language processing tasks and then a sequence model is needed in order to fit the methodology in that different context.

3.1.2 Interpretable Features Extraction

The *interpretable features extraction* is the most critical phase in that from the input it is attempted to selected some features that are related in order to understand which impact they have on the prediction.

The question that arise is, what are *features*?
Also features are closely tied on the type of input of the model:

- In case of *Tabular Data*, features can be a column o a set of columns
- In case of *Image Data*, features can be a subset of the image and then a portion of it that can be defined as a set of pixels.
- In case of *Text Data*, features can be a set of characters, a group of words or maybe also a sentence or a group of sentences.

The main problem is to define features that are *related*. For example in the case of images related features are all pixels that identify a distinct object, this is a complex process but there are in literature some techniques to do it. In EBANO on images features are extracted with segmentation techniques by extracting the values of some convolutional layers. In case of texts instead is complicated define what are related features because is not possible to extract distinct object but is needed to extract some concepts by grouping words. Some examples of related words can be: words inside the same sentence, or words with similar meaning or again words with similar vector representation but is more difficult to make these features very understandable by humans.

While for humans is relative easy to express related concepts, for machine is very difficult to automatically discover related concepts given some input data, specially for unstructured data.

3.1.3 Perturbation

A *perturbation* can be viewed as a *Noise* that is inserted in the original input in correspondence of the previous selected features. Also in this case the type of perturbation is closely related on the type of input. A possible type of noise can be remove the selected features and then remove the concept from the input, or change the feature with something else. For example in the case of images a possible perturbation technique can be to insert a blur in correspondence of the selected feature.

In case of input textual data instead, a possible perturbation can be change a character inside a word, or moreover removal or change entirely words or also remove entirely sentences.

3.1.4 The estimate of influences

To estimate the *influence* that a feature had for the prediction of the original input are taken two output of the model:

- *Original Prediction*: is composed by the original probabilities and class label predicted by the model given the original input.
- *Perturbed Prediction*: is composed by the new probabilities and class label that the model output by making again the prediction with the new input where it is applied a perturbation over a selected feature.

The underlying idea is that a feature impacts positively the prediction of a class label if adding a noise or a perturbation on that feature the probability of belonging to the class of interest is weakened, while a feature impacts negatively a prediction of a class label if after a perturbation on it the probability of belonging to the class is reinforced. The reason is that, if applying a perturbation on the feature extracted the probability decreases, this implies that it is decreased for the absence of the concept associated to the feature and then is influential for the analyzed class label.

For this purpose are defined two indices that quantify with a numerical values the influence of the interpretable feature over the class of interest:

- *nPIR* normalized Perturbation Influence Relation
- *nPIRP*: normalized Perturbation Influence Relation Precision

These two indices are a general approach independent from the task, type of input and type of neural network and then are taken from *EBANO* on image and adapted also to the text classification task.

normalized Perturbation Influence Relation

nPIR stands for *normalized Perturbation Influence Relation* and represents the impact of the interpretable feature selection for the class of interest. It represents with a numerical value if the feature has a positive or a negative impact for the prediction of that class. It takes into account the original predicted probabilities and the probabilities after the perturbation. If after the perturbation the probabilities of belonging to the class c are reduced then the feature will be positively influential for the original prediction, otherwise it will be negatively influential.

Let formalize the problem in mathematical terms, given:

- C the set of class of interests
- $ci \in C$ the class of interest
- $p_{o,ci}$ the original probability to belong to the class ci
- F the set of features extracted
- $f \in F$ the current feature
- $p_{f,ci}$ the probability to belong to the class ci after the perturbation f

Firstly is calculated the *Delta Impact*:

$$DI_f = p_{o,ci} - p_{f,ci} \tag{3.1}$$

It is a simple difference between the original probability for the class of interest ci and the probabilities of the input after the perturbation. Probabilities

are in domain [0,1], this implies that the *Delta Impact's domain* is [-1,1]. If DI_f is greater then 0, implies that original probabilities $p_{o,ci}$ is greater then the probabilities after the perturbation $p_{f,ci}$ and then the feature f positively influence the prediction of the class label. Otherwise, if $p_{o,ci}$ is minor then 0, the feature f has a negative impact on the prediction of the class label. With DI_f is expressed the intensity of the influence and to calculate the influence it is weighted by the *Symmetric Ratio Impact* that represent the relative impact of the perturbed feature *f*.

$$SRI_f = \frac{p_{o,ci}}{p_{f,ci}} + \frac{p_{f,ci}}{p_{o,ci}} \tag{3.2}$$

Then the *PIR* that stands for *perturbation influence relation* is calculated as follows:

$$\begin{aligned} PIR_f &= DI_f * SRI_f \\ &= (p_{o,ci} - p_{f,ci}) * (\frac{p_{o,ci}}{p_{f,ci}} + \frac{p_{f,ci}}{p_{o,ci}}) \\ &= p_{f,ci} * (1 - \frac{p_{f,ci}}{p_{o,ci}}) - p_{o,ci} * (1 - \frac{p_{o,ci}}{p_{f,ci}}) \end{aligned} \tag{3.3}$$

To optain the final *nPIR*, it is applied normalization over the *PIR* value:

$$nPIR = softsign(PIR) \tag{3.4}$$

where:

$$softsign = \frac{x}{1 + |x|} \tag{3.5}$$

The nPIR domain is [-1,+1] where +1 means that the feature f has a very high positive influence for the prediction of the class label *ci*, 0 means that the feature is neutral and -1 that has a very negative impact on the predicted class label.

normalized Perturbation Influence Relation Precision

nPIRP stands for *normalized Perturbation Influence Relation Precision* and represent the precision to which the model has learned a specific feature. This is due to the fact that a feature can impact more classes together, if a feature impacts a large number of classes implies that the concept represented by

the feature is general and then it is associated to different possible output pattern. In this case the feature has a negative precision; otherwise if the feature and its concept impact mainly the class of interest it will have a positive precision. Here are reported the mathematical formulas without going much in the details because for binary classification problem, that is the case of study of this thesis, is not useful.

$$\xi_{ci} = p_{o,ci} * |nPIR_{ci}| \tag{3.6}$$

$$\xi_{C/ci} = \sum_{c}^{C/ci} p_{o,c} * max(0, nPIR_c) \tag{3.7}$$

Then the *PIRP* that stands for *perturbation influence relation precision* is calculated as follows:

$$PIRP_f = DI_f(\xi_{ci}, \xi_{C/ci}) * SRI_f(\xi_{ci}, \xi_{C/ci}) \tag{3.8}$$

To optain the final *nPIRP*, it is applied normalization over the *PIRP* value:

$$nPIRP = softsign(PIRP) \tag{3.9}$$

Also in this case the domain ranges in [-1,+1] where +1 indicates that the feature impact only the current class label *ci*, -1 indicates the feature impact more the other classes with respect of the current class label and 0 indicates that the feature impact all the classes in the same way.

3.1.5 Local explanation

The local explanation of a given input and a predicted original label consists of a list of features selected from the input to which is applied separately a perturbation and for each perturbation it is provided a *Numerical Explanation* and a *Visual Explanation* that show which are the features belonging to the perturbation and how much they are influential for the original class label. For each new perturbed input are provided two type of explanation:

- *Numerical Exlpanation*: The numerical explanation is composed by the two scores indices *nPIR* and *nPIRP* discussed in the previous section. It expresses how much the feature impact the prediction of the class label, which are the amplitude of the impact and how much is precise.
- *Visual Explanation*: The visual explanation attempts to show in a visual representation which are the feature selected from the input and how much each one is influential.

3.2 EBANO Text Architecture

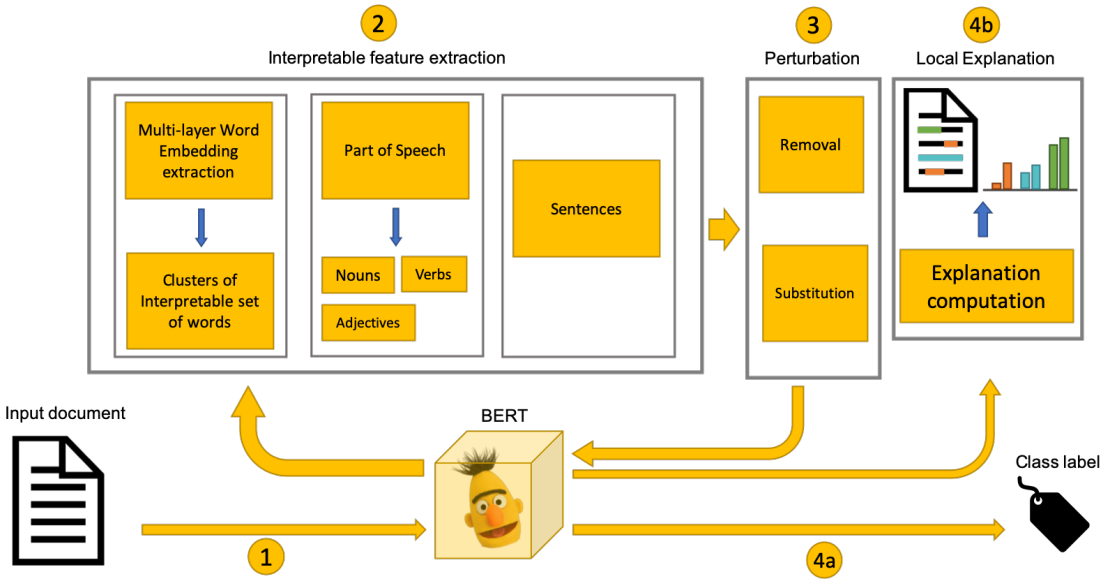

Figure 3.2. EBANO Text Architecture

In this section is punctually described the *EBANO Text architecture* to explain, given a *fine-tuned BERT model* for *sentiment classification* and an input text review to predict the output, how the black-box model made the prediction. In other words it is the architecture to provide the *outcome local explanation* of the predicted class label of a given input text, by applying different types of *perturbations* over different types of *interpretable features extracted*. The overall architecture and workflow is showed in figure 3.2 and is the specialization of the *General Methodology* 3.1 in the context of *Natural Language Processing*. The experimental choices about the task and the black-box model used will be motivated later in the section 4.1.

3.2.1 Input feeding

The input document containing the input review is feeded into the BERT model. The input is tokenized as wordpieces and it is representend with token embedding, segment embedding and position embedding like standard BERT's representation as explained in section 2.3.1. The input text is also pre-processed by removing html tags.

3.2.2 Interpretable features extraction

The *interpretable features extraction* is the most critical phase in the workflow. In this work, features are extracted at tokens and sentences granularity level while characters features extraction are not treated. All this features extraction methods are *positional* this means that same word in different positions are treated as different words. To do this each token is represented as a tuple *(id,word)*. With this representation is possible to exploit the potential of *BERT's contextualized word embeddings* 2.3.4.

Three different type of *interpretable features extraction* are possible in *EBANO Text*, each of them gives a different meaning to what are *related features*:

1. *Part-of-speech feature extraction*
2. *Cluster feature extraction*
3. *Multi-layer Word Embedding feature extraction*

a) Part-of-speech features extraction

This type of feature extraction, selects words that are *related* in the sense that *belong to the same part-of-speech*. The intuition below this type of features extraction is that different parts-of-speech can have different meaning for the prediction and the model needs to be able to capture this semantic difference. Tokens are divided in groups like adjective, noun, verb, adverb and so on.
After the tokenization phase, it is attached to each *(id,token)* tuple an additional field containing the part-of-speech tag to which it belong to. This implies that each word is represented as a 3-Tuple *(id,token,tag)*.

Then it is created a list of list:

- Each element of the *outer list* corresponds to a different part-of-speech tag. There is an outer list for each part-of-speech taken into account and each of them will be a different feature selection where, by applying a perturbation, will be created a new perturbed text.
- Each *inner list* contains the list of *(id,token)* tuple corresponding to the current part-of-speech tag. In other words this list contains all words and their relative position that belong to the current feature selection.

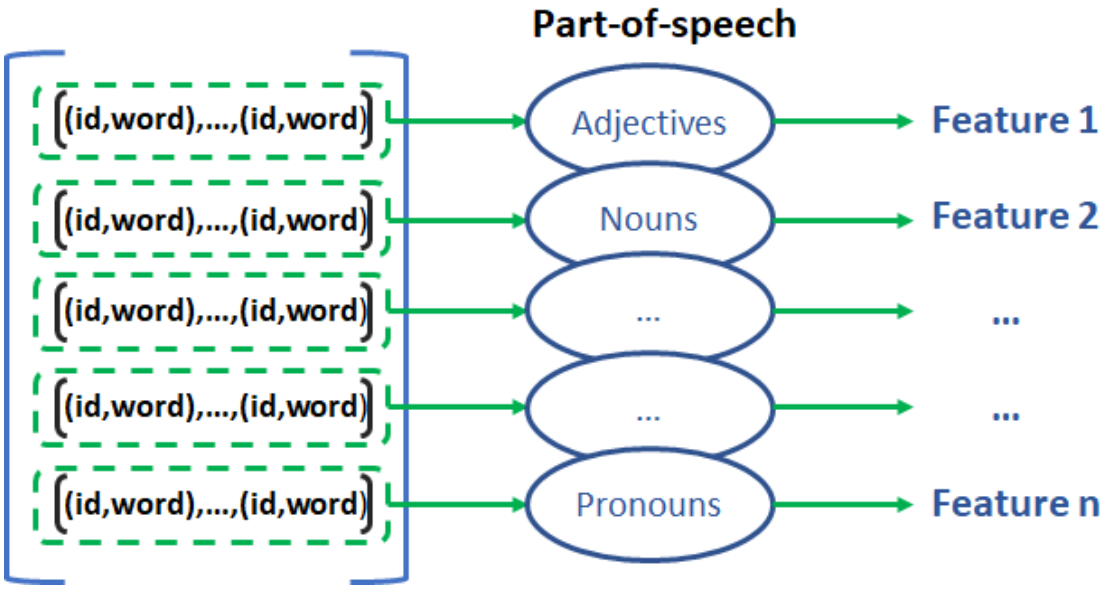

Figure 3.3. Part-of-speech feature extraction

b) Sentence features extraction

This type of feature extraction instead, selects words that are *related* in the sense that *belong to the same sentence in the input text*. The intuition below this type of features extraction is instead that whole sentences can have a complete meaning and the model need to be able to capture it. This time, after the tokenization phase, it is attached to each *(id,token)* another number that corresponds to the id of the sentence to which it belongs in relation to the whole input text. In this case the 3-Tuple is *(id,token,sentenceId)*.

Also in this case it is created a list of list:

- Each element of the *outer list* this time corresponds to a different sentence in the whole text. There is an outer list for each sentence in the input text and each of them will be considered as a different feature selection to which apply a perturbation to create the new input texts.

- Each *inner list* contains the list of *(id,token)* tuple belonging to the current sentence.

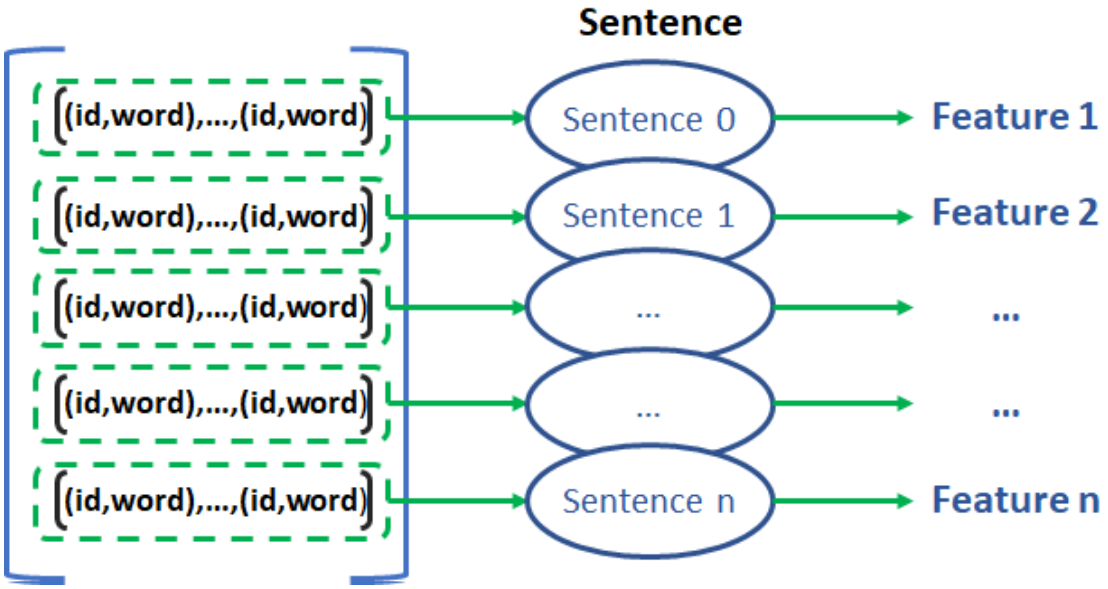

Figure 3.4. Sentence feature extraction

c) Multi-layer word embedding features extraction

In this case words are grouped together by looking their featurized representation given by the model. This implies that the meaning of *related* in this case is that have *similar word embedding representation*. The intuition is that words with a similar representation learned by the model probably have similar meaning and then can be influential if grouped together. To do this it is applied the *k-means clustering* algorithm over the word embedding of each word to figure out groups of related words. In this case the list of list is composed as follows:

- Each element of the *outer list* corresponds to a different cluster. There is an outer list for each cluster that correspond to a different feature selection and then to a new perturbed text.
- Each *inner list* contains the list of *(id,token)* tuple corresponding to words belonging to the current cluster.

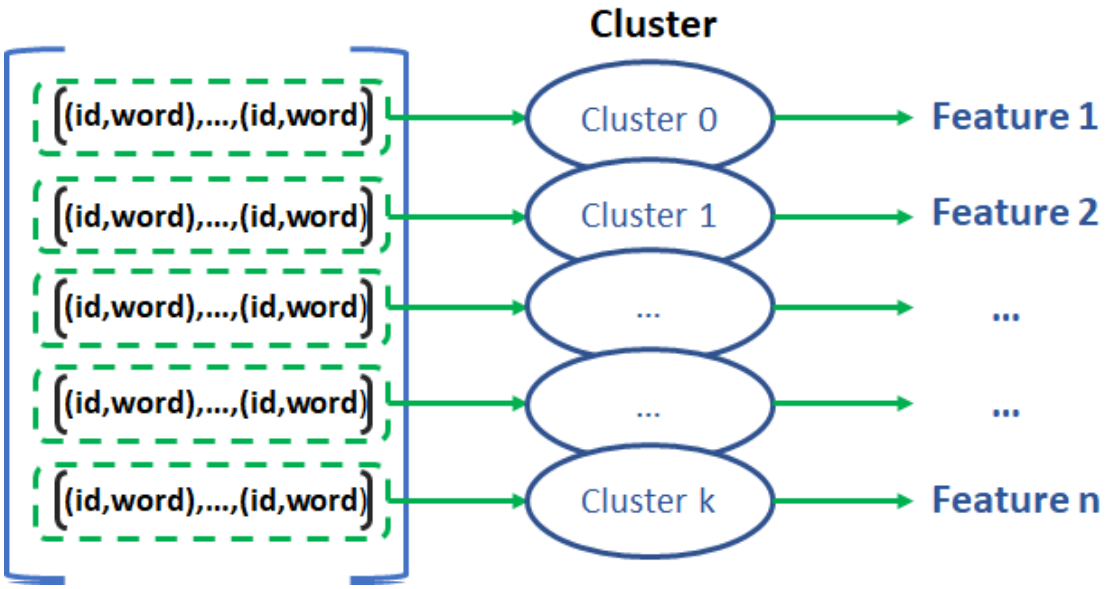

Figure 3.5. Clustering feature extraction

Features extraction methods comparison

The *part-of-speech feature extraction* method tries to explore the semantic meaning of words by looking to which different part-of-speech they belong to. The *sentence feature extraction method* method instead explores more the position meaning learned by the model for each word. Finally instead the *multi-layer word embedding feature extraction* method is the only one that look really inside the model to figure out which kind of representation it has given to words by looking the values of the lasts transformer encoder layers. This last feature extraction method explores the learned features of the fine-tuned model to figure out what it has learned and also their context. The types of feature extraction are three because, looking different aspects of the input text, can provide different types of explanation.

Comparing instead the execution time, for the part-of-speech and sentence feature selection is near to zero in that the tagging phase is not much time consuming, while the feature extraction phase for the multi-layer word embedding feature extraction is very more heavy.

3.2.3 Perturbation

After selecting a set of interpretable features, as explained in the previous part, *EBANO Text* applies a *perturbation* in correspondence of these features. A perturbation can be viewed as a noise that is added to the input text to create a new perturbed text without the selected concept. The input of this phase is the list of list from the previous phase and the output is a list of perturbation, and then a list of new perturbed texts, for each element in the outer list in input.

Two different types of perturbation are proposed:

1. *Removal perturbation*
2. *Substitution perturbation*

a) Removal perturbation

All the selected features are *removed* from the input text. Multiple blank spaces are merged into one and spaces before punctuation are removed to produce a new text that is written like humans. One new perturbed text is created for each of the outer list and correspond to a different perturbation.

Example of a single removal perturbation:
INPUT TEXT: This is a good **example** of removal **perturbation**
PERTURBATION: This is a good ~~example~~ of removal ~~perturbation~~
PERTURBED TEXT: This is a good of removal

b) Substitution perturbation

For each of the selected features looking for a possible antonym of the word, if one is founded the original word is *substituted* with the antonym, otherwise it will remain unchanged. Also in this case is produced a new perturbed text that correspond to a different perturbation for each of the outer list.

Example of a single substitution perturbation:
INPUT TEXT: Now a **good** example of substitution perturbation
PERTURBATION: Now a ~~good~~ "bad" example of substitution perturbation
PERTURBED TEXT: Now a **bad** example of substitution perturbation

3.2.4 Local explanation

Feeding the *Input Original Text* into the model, are extracted:

- Original Predicted Label
- Original Probabilities
- Original Label

Each *Perturbed Text* is also feeded into the model and are extracted:

- Perturbed Predicted Label
- Perturbed Probabilities

The *Original Label* is extracted only if is available from the input dataset otherwise it is labeled as [Unknown]. This label is important to understand if, with the original text, the model takes a good or a wrong prediction.

Given the *Original Predicted Label* and the current *Perturbed Predicted Label* it is possible to observe if the current perturbation has changed the prediction of the class label. Furthermore comparing the *Original Probabilities* and the *Perturbed Probabilities* it is possible to observe how much this current perturbation has changed the prediction (not necessarily changing the predicted class label). Due to this purpose are calculated the two indices explained in the section 3.1.4:

- nPIR - normalized Perturbation Influence Relation
- nPIRP - normalized Perturbation Influence Relation Precision

All this information are saved in a json file called *Transparency Report* that allows the final user, with some queries, to visualize all information desired. It is possible to filter by choosing the type of features extraction, the type of perturbation or the value of influence indices.

Local explanation

The *transparency report* is the responsible for providing the *Local Explanation* of the input text and the outcome of the model. In the first section are report all the information about the original text and the original outcomes of the model. The information reported are the following:

- *Original Input Text*
- *Original Label*
- *Original Predicted Label*
- *Original Predicted Probabilities*

The second section is divided in different subsection for each couple of *feature extraction* and *perturbation* type created. Each of these subsection is composed by a list of perturbations, where each perturbation contains the following information and provides the *visual explanation* 3.6 and the *numerical explanation* 3.7:

- *Features selected*
- *Text Before Perturbation with highlighted features*
- *Text After Perturbation*
- *Perturbed Predicted Label*
- *Perturbed Predicted Probabilities*
- *nPIR and nPIRP*

In the following figures 3.6, 3.7 is showed an example of a single perturbation over a single feature extracted. The full report will contains a list of them, where each one corresponding to a different perturbation over a different feature extraction. The objective of this figure is to show which are the information reported in the final output.

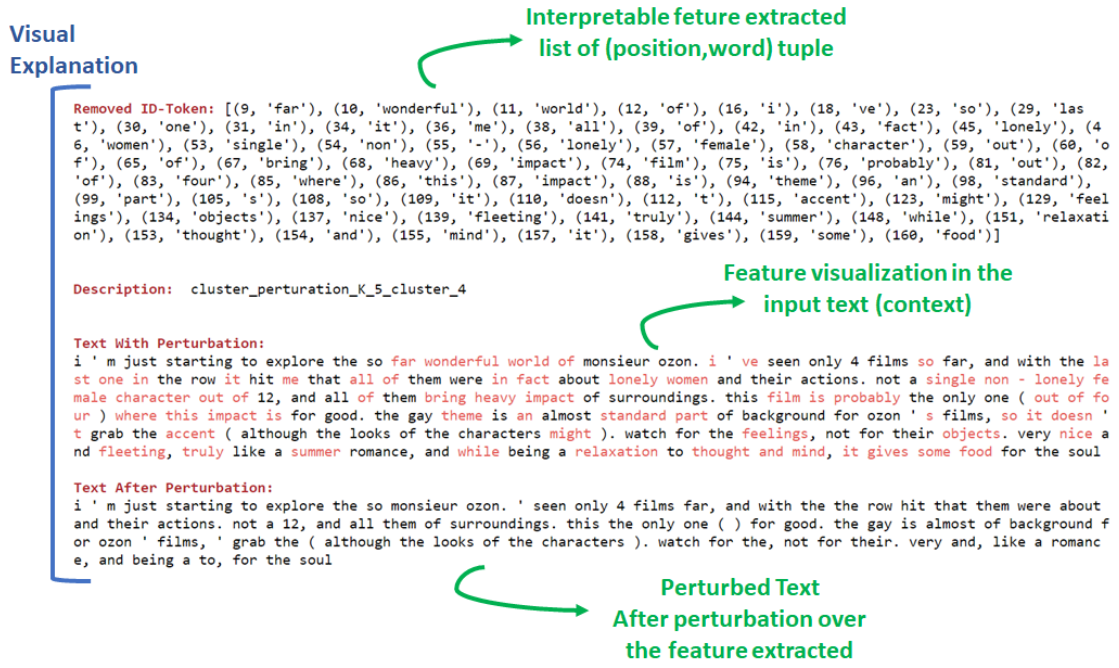

Figure 3.6. Local explanation: Visual explanation.

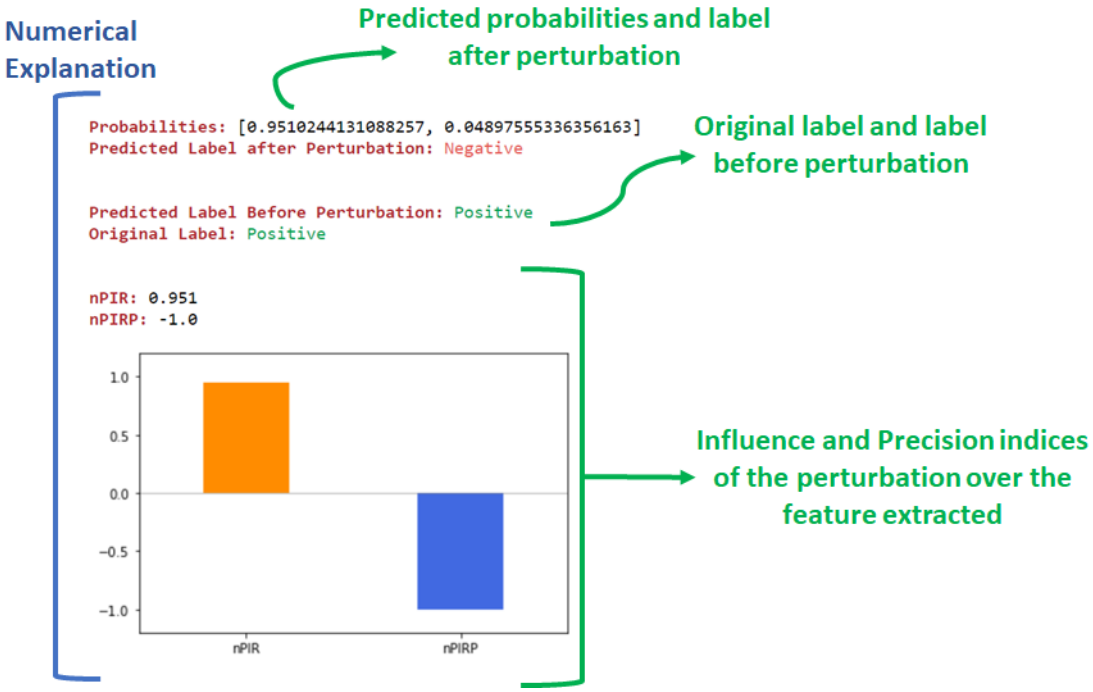

Figure 3.7. Local explanation: Numerical explanation.

3.2.5 Parts-of-speech features extraction details

Each part-of-speech selection is a macro categories containing sub-tags. For example the adjective part-of-speech selection is the union of standard adjectives, adjectives comparative, adjectives superlative and so on. This union is coherent with the grammar point of view. Apart from this are created other macro categories that are not closely related to grammar as the "Others" part-of-speech selection that contains foreign words, symbols, number and so on. These groups have the aim of reducing the complexity of the problem and making human understandable the features selection. Indeed having a lot of parts-of-speech categories will create more perturbations and more likely less influential. Another reason of this choice is that reducing the number of macro categories is easier to create perturbation where the feature selection is the combination of these categories.

More precisely the macro categories defined are:

- *Adjectives*: all type of adjectives, comparative, superlative etc.
- *Nouns*: all type of noun, singular, plural etc.
- *Verbs*: All type of verbs
- *Adverbs*: contains the union of all types of adverbs and modals.
- *Conjunctions*: union of conjunctions, prepositions and determiners
- *Pronouns*: union of all types or pronouns and all the WH words like what,which etc.
- *Interjection*: all type of interjections
- *others*: contains foreign words, symbols, numbers

3.2.6 Multi-layer word embedding features extraction details

This features extraction method tries to find similar words by clustering the inner representation learned by the model during pre-training and fine-tuning. The model is composed by multiple transformer encoder [23] layers and each of them gives a possible featurized representation of the word. In the base version of BERT there are 12 layers each one of 768 dimensions while in the large version there are 24 layers each one of 1024 dimensions.

In the *Multi-layer Word Embedding feature extraction* method are extracted all features of the last four layers of the model, that in this case of study is the base version, for each wordpiece. This implies that is extracted an *Embedding Tensor* of shape ($N°_{Wordpieces}$, $N°_{Features} = 768$, $N°_{Layers} = 4$) and then each wordpiece is represented by a matrix of shape ($N°_{Features} = 768$, $N°_{Layers} = 4$).

The *clustering algorithm* is not applied directly on the tensor but firstly are applied a set of *transformations* as showed in figure 3.8.

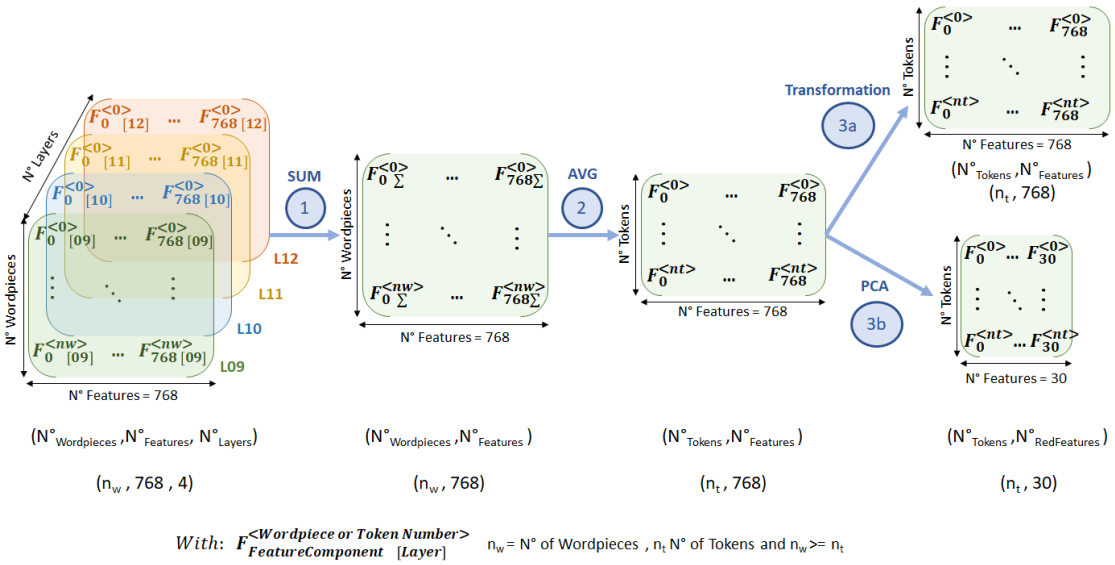

Figure 3.8. Word Embedding transformations.

Word embedding transformations

1. **SUM:** Starting from the tensor read from the model each component of each wordpiece embedding is summed over the layer axis as showed in equation 3.10 . The wordpiece's representations passes from the matrix of shape (N°features,N°Layers) to a single vector of shape (N°Features).

   The sum has the purpose of maintain a good representation of the wordpiece but reducing the dimensionality over one axis.
   For each feature vector of each wordpiece it is applied:

$$F_{sum}^{<i>} = \sum_{l=1}^{4} F_{[l]}^{<i>} \tag{3.10}$$

   Where $F_{[l]}$ is the feature vector for the layer l and i is the wordpiece.

   This transformation reduces the size of the representation from the tensor of shape ($N_{Wordpieces}$, $N_{Features}$, N_{Layers}) to an embeddin matrix of shape ($N_{Wordpieces}$, $N_{Features}$).

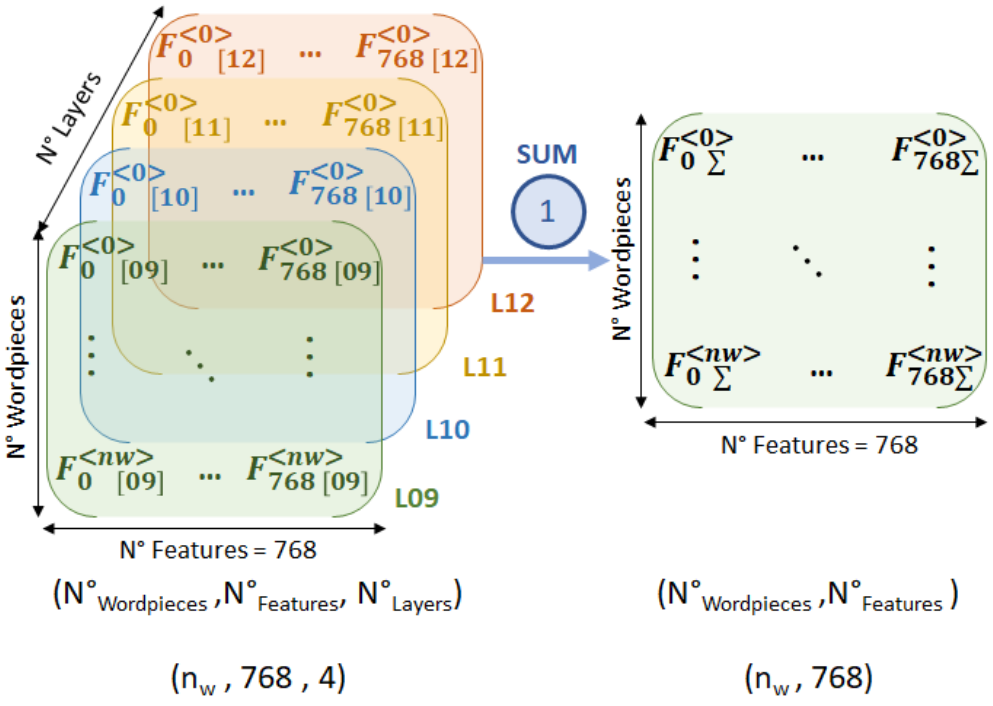

Figure 3.9. SUM Word Embedding transformation.

2. **AVG:** features extraction it is applied at token granularity, but BERT's input is splitted into wordpieces and for each of them are extracted the features. To join wordpieces corresponding to the same token it is applied the averaging of each component for each wordpiece.
For each tokens splited into wordpieces is applied:

$$F_{tk} = \frac{\sum_{w=1}^{nw} F_{sum}^{<w>}}{nw} \tag{3.11}$$

Where F_{tk} is the feature vector of the token *tk*, *nw* is the number of wordpieces to which the token is splitted and $F_{sum}^{<w>}$ is the vector representation, summed over axis, of each wordpiece *w*.

This transformation reduces the size of the representation of the embeddin matrix from shape ($N_{Wordpieces}$, $N_{Features}$) to (N_{Tokens}, $N_{Features}$) because is valid the following equation:

$$N_{Tokens} \leq N_{Wordpieces} \tag{3.12}$$

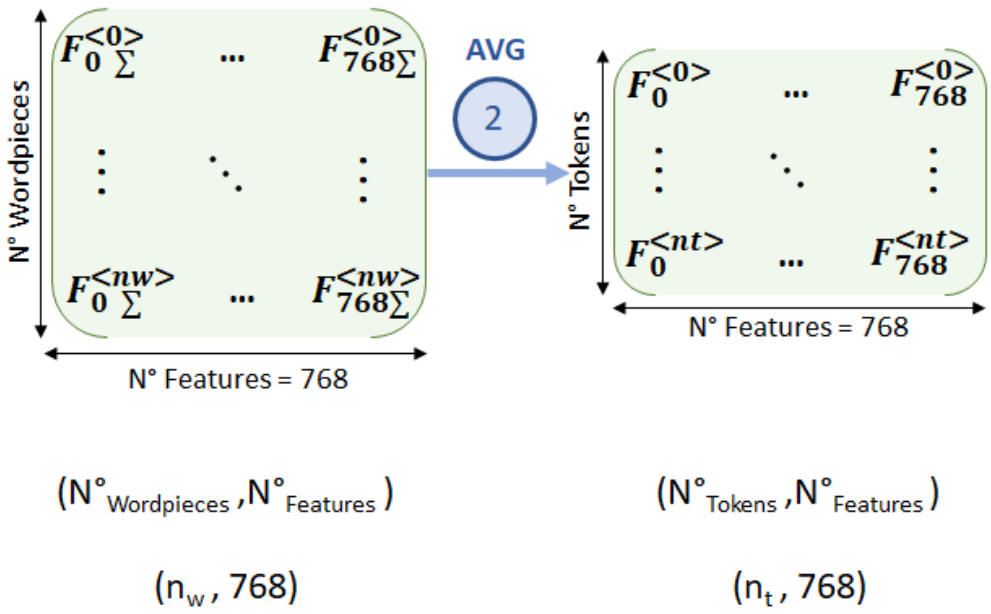

Figure 3.10. AVG Word Embedding transformation.

3. Before applying the clustering algorithm are applied also a set of pre-processing techniques as showed in figure 3.11:

   - **a) pre-processing**: it is used to standardize features by removing the mean and scaling to unit variance or normalize samples individually to unit norm. This kind of transformation keeps unchanged the embedding matrix representation as (N_{Tokens}, $N_{Features}$) where $N_{Features}$ is equal to 768.

   - **b) PCA**: it is used to reduce features dimensionality by applying the PCA algorithm. Clustering on high-dimensional data is known problem in data mining. To solve this are trained two PCA models one reducing to 30 features and the other reducing to 50 features. The PCA models are trained by taking random contextualized word embedding words from random input review taken from the test set of the dataset in order to maintain the same distribution of the training dataset. This trained models are saved into local disk and are used for reduce contextualized word embedding of new words. It is experimented that reducing to 30 features works better.

     This kind of transformation reduces the embedding matrix representation shape from (N_{Tokens}, $N_{Features}$) where $N_{Features}$ is equal to 768 to (N_{Tokens}, $N_{F\_Reducted}$) where $N_{F\_Reducted}$ is equal to 30.

   This two alternatives,**3a** and **3b**, have the objective of improving the performance of the cluster algorithm. Comparing the results to observe which is better is not easy due to high-dimensional of input data but, looking lot of example, the PCA seems to perform better as it divides clusters into more homogeneous groups of words.

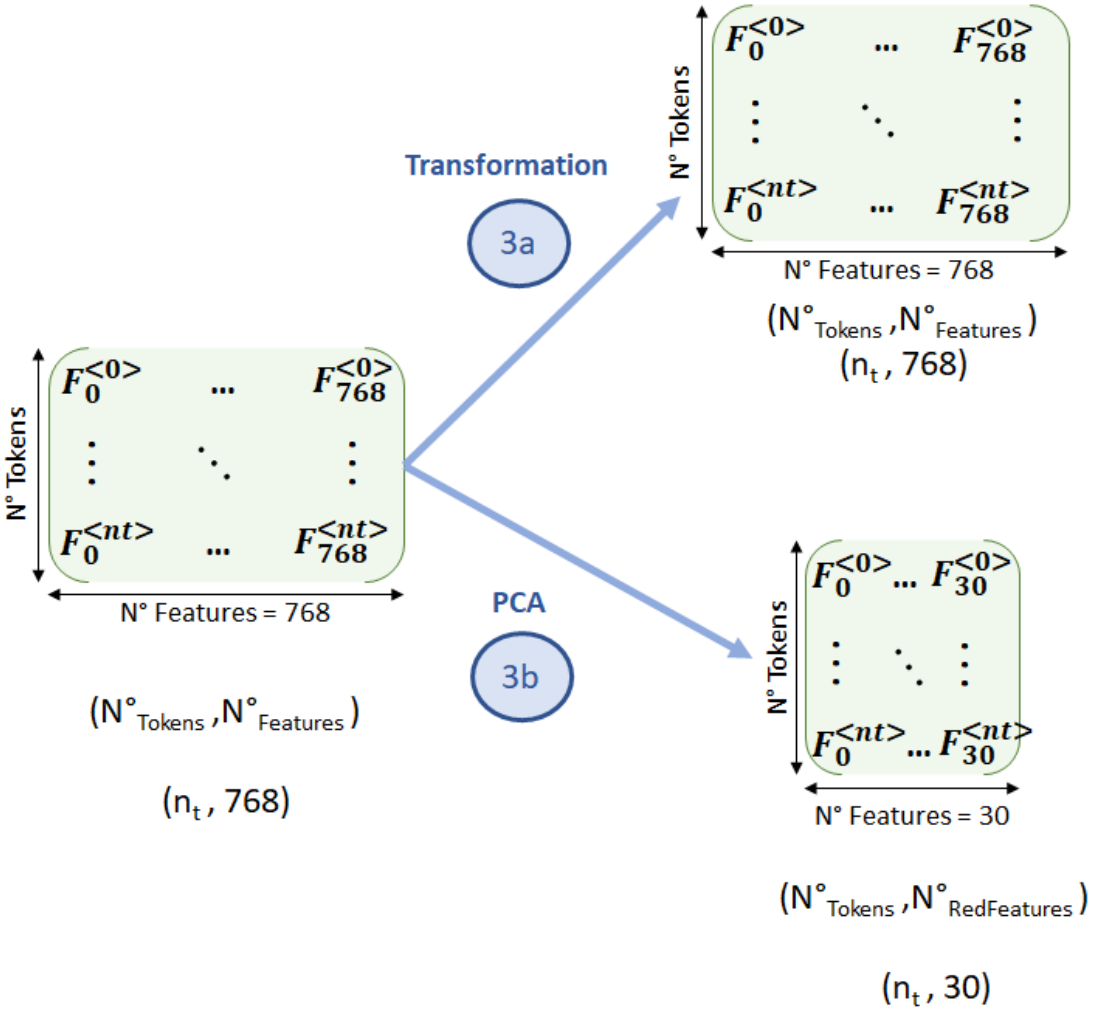

Figure 3.11. PCA Word Embedding transformation.

Clustering algorithm

Once extracted word embedding and have applied the set of transformation until reaching the final embedding matrix, it is possible to clusters features to get similar groups of tokens. The algorithm used to get clusters of similar words by their multi-layer representation inside the model is the *K-Means* algorithm 2.1.1. The hyperparameters of K-Means are selected with a large number of tests. The final values are:

- Maximum number of iterations = 5000
- Different initial centroids = 150
- Initialization of centroids = 'random'

This choice is due to have a good tradeoff between performance and computational time. The most important parameter of the k-means algorithm is the number of clusters K. Due to the high dimensionality of word embedding it is difficult to have a priori understanding of the distrubution of data and then the k-means algorithm is runned for different values of K and is chosen the *top-K* that gives a better division of words as showed in figure 3.12.

Top-K evaluation

The clustering algorithm is applied for different values of K. The minimum number of possible clusters is 2 while the maximum number of clusters cannot be fixed but needs to be function of the size of the input text. For this reason the K_{max} is calculated as follows:

$$K_{max} = \sqrt{N_{words} + 1} \tag{3.13}$$

This implies that the k-means is runned with values of K that are in range [2,Kmax] and after are all evaluated to understand which one givea a better division of words.

K score

To each possible K as number of clusters, it is assigned a K *score*.
The K *score* is calculated as follows:

$$K_{score} = \max_{c=0}^{K} \left(\frac{nPIR_c}{len(c)} \right) - \min_{c=0}^{K} \left(\frac{nPIR_c}{len(c)} \right) \tag{3.14}$$

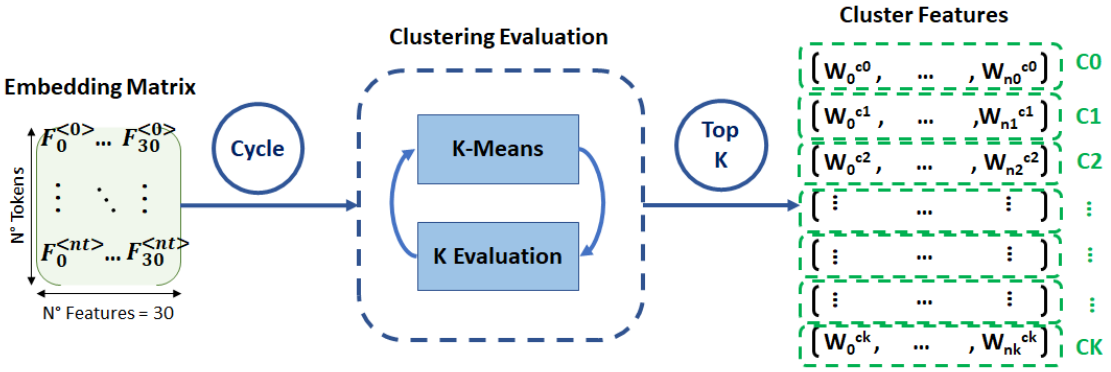

Figure 3.12. Clustering evaluation.

Where c is the current cluster, $len(c)$ is the number of words inside the current cluster and $nPIR_c$ is the nPIR value calculated with a perturbation over the current cluster c.

Given K clusters, is calculated for each cluster c the *Weighted nPIR*, weighted by the number of words belonging the cluster. This is due to reward the smaller cluster of words that is more influential for the prediction. The behind intuition is that big clusters of words are more likely to contains influential words and with this score calculation is attempted to find the smaller cluster of words that is high influential. Finally the K_{score} is simply calculated as the difference between the *max* and the *min* of the *Weighted nPIR* calculated before for each cluster c.

Also other methods have been tried like the calculation of the *Silhouette* as explained in 2.1.1, but the proposed method is function of the objective of this clustering that is not to divide in the best way words, but is to divide words in group that are more influential. Once founded the K with the highest score, it is defined as top-k and their relative clusters are taken as output of the feature extraction phase.

In the figure 3.13 is is possible to see the entirely workflow to obtain the cluster feature extraction starting from the input text and all the transformation of the embedding representation before applying the k-means algorithm.

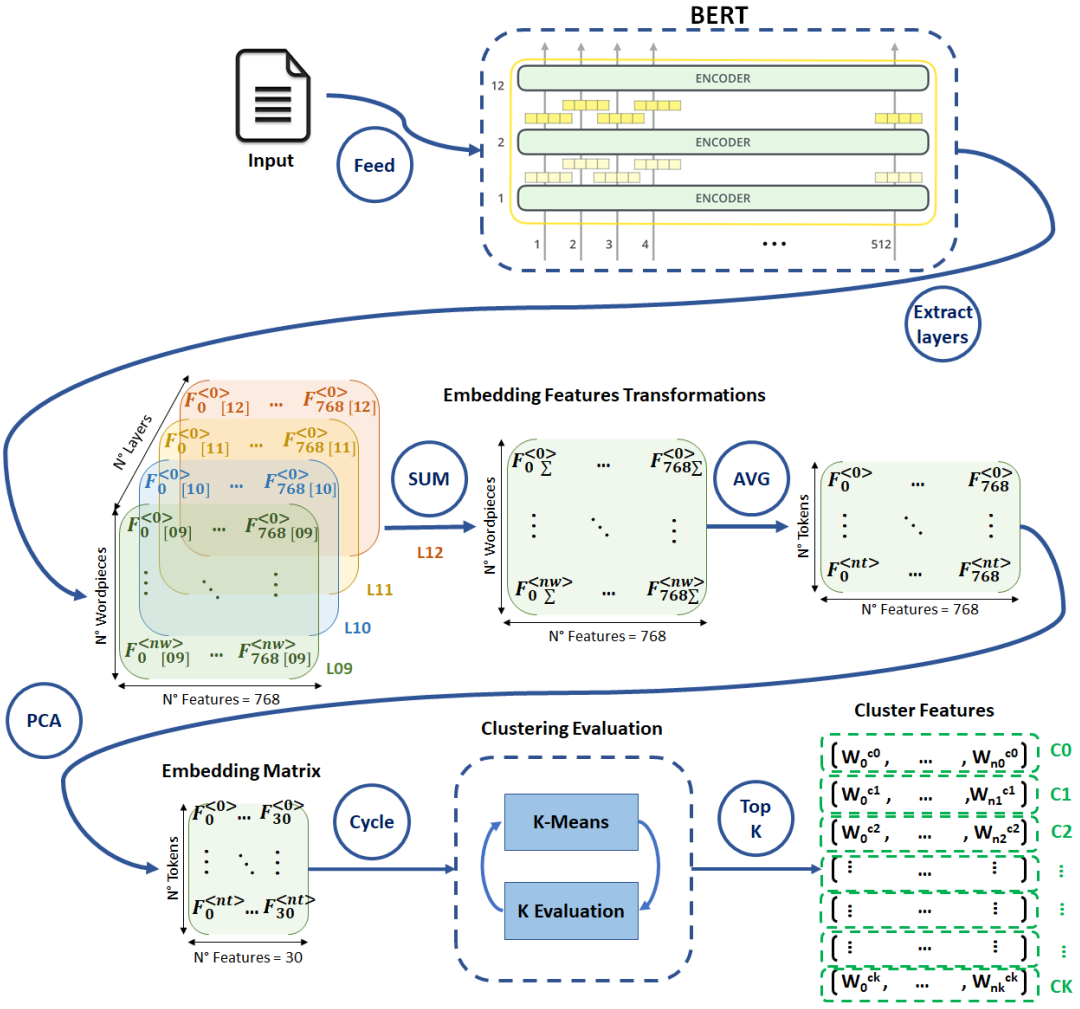

Figure 3.13. Multi-layer Word Embedding feature extraction full schema

Chapter 4

Experimental Results

In this chapter will be motivated all the experimental choices in order to apply the proposed methodology in the context of natural language processing. In particular are motivated the choices of the dataset, the deep neural network used and the experimented task. Moreover are showed two examples of local explanation of input text individually highlighting the salient aspects of the process of explanation. The first example shows a short review in order to understand with a simple and practical example how the methodology works, the second example instead shows how is possible to provide a meaningful *local explanation* also with complex input text. A part of this are reported some considerations and some global results that shows how it is possible to analyze each class label globally by looking all the influential feature in a large corpus of input texts.

4.1 Experimental settings

In the following section are explained all the choices to apply the *General Methodology* explained in the chapter 3.1 in the context of *Natural Language Processing* explained instead in the section 2.2.

4.1.1 Task

The experimented task is *sentiment analysis*. This is because, due to the complex nature of text and natural language tasks, is more easy to start with a binary problem (predict if the input review is positive or negative). The objective is to experiment the methodology, independently from the chosen task, then staring from an easy task and after try to generalize the methodology. Another reason for choosing this task is that is a well know problem and there is a lot of documentation and code on the web. The scores indices explained before (nPIR and nPIRP) is a general approach for multi-class classification. Due to the fact that in this case it is a binary classification problem, the precision index nPIRP is not significant but it is left there because the approach can be extended to a very similar multi-class task like topic detection.

4.1.2 Dataset

The chosen dataset is the *IMDB reviews dataset* [9]. It is composed by 50000 film's review divided in positive and negative equally. In the filename it is specified the id of the review and the score in the format:

- *reviewId_score*

Where score is a number in the range [0,10].

- 0 means that the score of the review is unknown.
- 1-5 means that the score of the review is negative.
- 6-10 means that the score of the review is positive.

Also in this case, it has been chosen because is a very well know dataset, with a lot of documentation available on the web and moreover is also one of the dataset contained in the TensorFlow framework.

4.1.3 Neural Network architecture

The neural network architecture chosen is *BERT* [4] that, as explained in section 2.3, is currently the state-of-the-art for many natural language processing tasks. It is used the smaller version of the pre-trained models (*BERT-Base, Uncased*) for performance and complexity reasons. The hardware used to all experiments is a normal PC accessible to normal people with a quad core cpu and 16gb of ram. Also in this case the scope is to experiment the methodology and after maybe generalize with a larger version of BERT and a more powerful hardware.

4.1.4 BERT fine-tuning

Starting from the *BERT-Base and Uncased* pre-trained model, it is fine-tuned for the *sentiment classification task*. To do this it is added a classification layer where, taking the representation of the whole sentence given by the [CLS] token and feeding it into a softmax, predict if the class label is (0/1) and then positive or negative. The model is trained with 25000 input reviews from the IMDB dataset, 12500 positive and 12500 negative, reaching an accuracy of 86%. The model is saved as Tensorflow estimator to be used again. The fine-tuning phase, exploiting transfer learning, is inexpensive compared with pre-trainig and then requires less computational power and less training data. With actual performance it is possible to fine-tune the model with about 14 hours in a local cpu and in a less then 1 hour in a Google Colab TPU.

4.2 Tools, frameworks and libraries

In this sections are reported the principal tools and libraries used and is explained where they are mainly used in order to realize the code of *EBANO Text*.

NLTK

NLTK [8] stands for Natural Language ToolKit and is a suite of libraries and software for human natural language processing. In EBANO is used maily for the following tasks:

- *Tokenization*: is used to split the input text into words and punctuation. This type of tokenizer is different from the one of BERT explained in section 2.3.1 because not works at wordpieces level but a tokens level.
- *Parts-of-speech tagging*: it is used to attach to each token the part-of-speech tag to which it belongs for the part-of-speech features selection 3.2.2.
- *Senteces tagging*: similar to part-of-speech, it is used the sentences tokenizer to split each input into sequence in order to create the sentence feature selection 3.2.2.

Numpy

Numpy [14] is a python library specialized in fast matrix operation or multi-dimensional strucutres. It is used for all the computational treatment of tensor, matrix and vectors.

Scikit-learn

Scikit-learn [15] contains a set of tools for data mining, data analysis and machine learning and it is builted on Numpy, Scipy and Matplotlib. In EBANO it is used mainly for the multi-layer word embedding feature extraction 3.2.6:

1. Pre-processing: during word embedding transformation are applied a set of preprocessing techniques like standard scaling and normalization.
2. PCA: always in the word embedding transformation it is used the pca tool to reduce the features of the embedding matrix. A pca model was trained and saved in local disk to be loaded again for new dimensionality reduction always with the same library.

3. K-Means Clustering: the clustering algorithm used for the multi-layer word embedding feature extraction it is applied with the k-means tools of scikit-learn.

Tensorflow

TensorFlow [10] is an end-to-end machine learning platform. It is open source and has a large set of tools and libraries. Moreover there is a lot of documentation and works on it available on the web. In this work tensorflow is used to everything about the neural network model and in particular to create the BERT model, to train it, to save the estimator of the model on local disk and to load the estimator to make new prediction on demand.

4.3 Examples of local explanation

In this section will be analyzed two different examples of input text in order to show the different feature extraction methods, perturbation methods and local explanations. The first example shows a simple review in order to make more understandable how the theory ideas are applied to provide the local explanation. The second one instead shows how the proposed framework works on a real large input review. Here are presented only two examples but the methodology was tested on a very large and various corpus of input texts.

4.3.1 Example of short review

In this first example is analyzed a very short review. It is very human understandable review and also simpler to predict for the model. This implies that can be a very good example in order to illustrate how the methodology works.

Original prediction

In the first part are reported the information about the *input review*, the *original label* expected, the *predicted label* and the *predicted probabilities* as showed in figure 4.1. This review is expected to be *negative* and the model correctly predict the class label with high probabilities.

```
Input Sentence:
This film was very awful. I have never seen such a bad movie.

Original Probabilities: [0.999329686164856, 0.0006703310064040124]

Original Label: Negative
Predicted Label: Negative
```

Figure 4.1. Example 1: Original prediction

Part-of-speech feature extraction and removal perturbation

Firstly, over the *part-of-speech feature extraction*, are applied a set of *removal perturbation*.

What emerges by applying this type of perturbation over this feature extraction method is that *adjectives* are very influential for the prediction as showed in figure 4.2, while *nouns* and *verbs* are not influential as showed in figures 4.3 and 4.4. All the others part-of-speech are not reported but are not influential for the prediction.

Analyzing these results emerges that *part-of-speech feature extraction* can provide as explanation both the words responsable of the predicted label, that in this case are *"bad"* and *"awful"*, and the different *parts-of-speech* influential for the prediction, that in this case are *adjectives*. both this aspects can be interesting to be analyzed in order to understand the behaviour of the model.

In this case the behaviour of the model is how a human expects because is easy to understand that these two words are the most important in the review, but is needed to verify that the model has taken the prediction exactly for these and not for other aspects.

This is the first part of the *local explanation*, each feature extraction looks different aspects of the input and then can be used together to provide a good explanation of the behaviour of the model.

```
Removed ID-Token: [(4, 'awful'), (12, 'bad')]

Description:  Adjective perturbation

Text With Perturbation:
This film was very awful. I have never seen such a bad movie.

Text After Perturbation:
This film was very. I have never seen such a movie.

Probabilities: [0.0016094991005957127, 0.9983905553817749]
Predicted Label after Perturbation: Positive

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: 0.998
nPIRP: -1.0
```

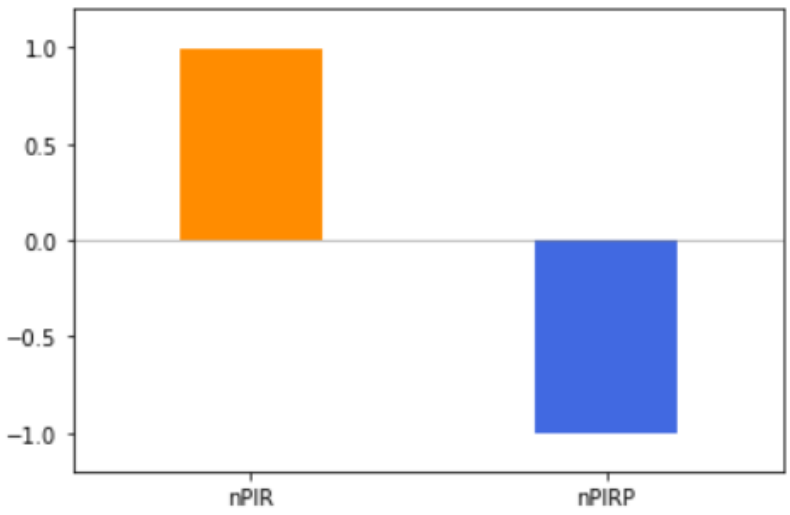

Figure 4.2. Example 3: Adjectives removal perturbation

Removed ID-Token: [(1, 'film'), (13, 'movie')]

Description: Noun perturbation

Text With Perturbation:
This film was very awful. I have never seen such a bad movie.

Text After Perturbation:
This was very awful. I have never seen such a bad.

Probabilities: [0.9979239702224731, 0.002075962955132127]
Predicted Label after Perturbation: Negative

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: 0.003
nPIRP: -1.0

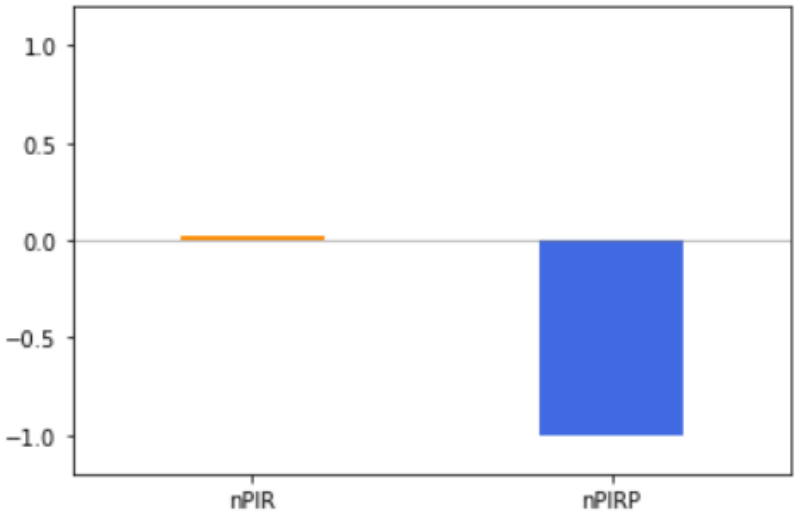

Figure 4.3. Example 3: Nouns removal perturbation

Removed ID-Token: [(2, 'was'), (7, 'have'), (9, 'seen')]

Description: Verb perturbation

Text With Perturbation:
This film was very awful. I have never seen such a bad movie.

Text After Perturbation:
This film very awful. I never such a bad movie.

Probabilities: [0.9991707801818848, 0.0008292430429719388]
Predicted Label after Perturbation: Negative

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: 0.0
nPIRP: -1.0

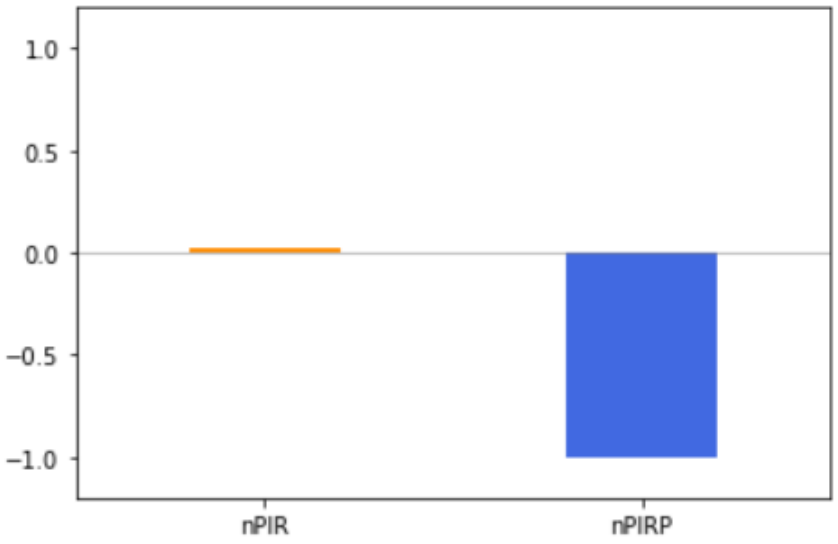

Figure 4.4. Example 3: Verbs removal perturbation

Part-of-speech feature extraction and substitution perturbation

Now, over the same *part-of-speech feature extraction*, are applied a set of *substitution perturbation* where each word is substituted with its antonym if founded. In this case, instead of analyze the absence of a concept, is analyzed the substitution with the opposite concept in order to denote the behaviour of the model.

Firstly, in figure 4.5, is reported the *substitution* over *adjectives* and what emerges is that, replacing the adjectives with their antonyms, the prediction changes. This is reasonable also for humans and the model captures it correctly.

In the figures 4.6 and 4.7 are reported the substitution over *verbs* and *adverbs* that, like all the other parts-of-speech not reported here, are not influential for the prediction.

One interesting point in order to obtain a *local explanation* can be to compare the *removal perturbation* and *substitution perturbation* over the same words.

```
Substituted ID-Token-Antonym: [(4, 'awful', 'nice'), (12, 'bad', 'good')]

Description:  Adjective substitution with antonyms perturbation

Text With Perturbation:
This film was very ("awful") nice. I have never seen such a ("bad") good movie.

Text After Perturbation:
This film was very nice. I have never seen such a good movie.

Probabilities: [0.001413671881891787, 0.9985862970352173]
Predicted Label after Perturbation: Positive

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: 0.999
nPIRP: -1.0
```

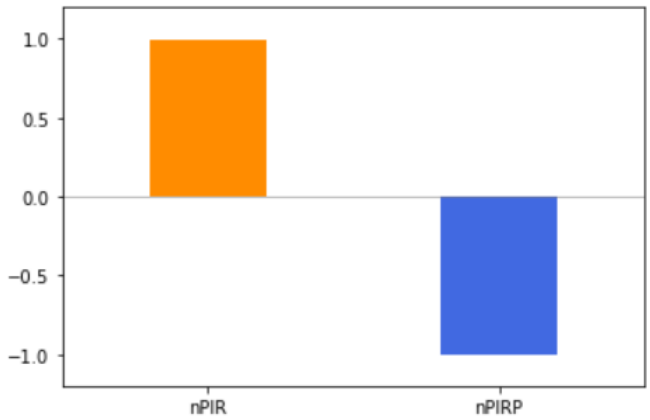

Figure 4.5. Example 3: Adjectives substitution perturbation

Substituted ID-Token-Antonym: [(2, 'was', 'differ'), (7, 'have', 'lack')]

Description: Verb substitution with antonyms perturbation

Text With Perturbation:
This film ("was") differ very awful. I ("have") lack never seen such a bad movie.

Text After Perturbation:
This film differ very awful. I lack never seen such a bad movie.

Probabilities: [0.9992532134056091, 0.0007467241957783699]
Predicted Label after Perturbation: Negative

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: 0.0
nPIRP: -1.0

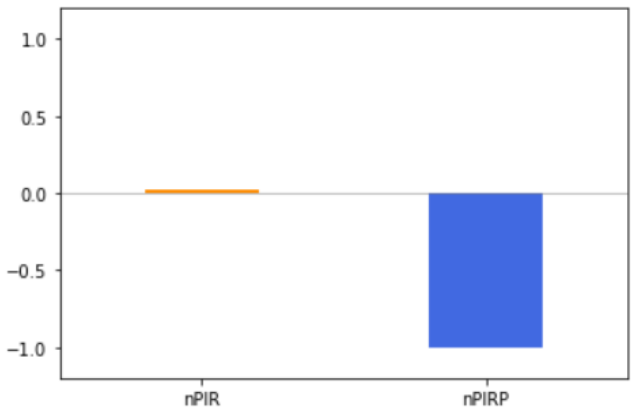

Figure 4.6. Example 3: Verbs substitution perturbation

Substituted ID-Token-Antonym: [(8, 'never', 'ever')]

Description: Adverb substitution with antonyms perturbation

Text With Perturbation:
This film was very awful. I have ("never") ever seen such a bad movie.

Text After Perturbation:
This film was very awful. I have ever seen such a bad movie.

Probabilities: [0.9993336796760559, 0.0006662862724624574]
Predicted Label after Perturbation: Negative

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: -0.0
nPIRP: 0.012

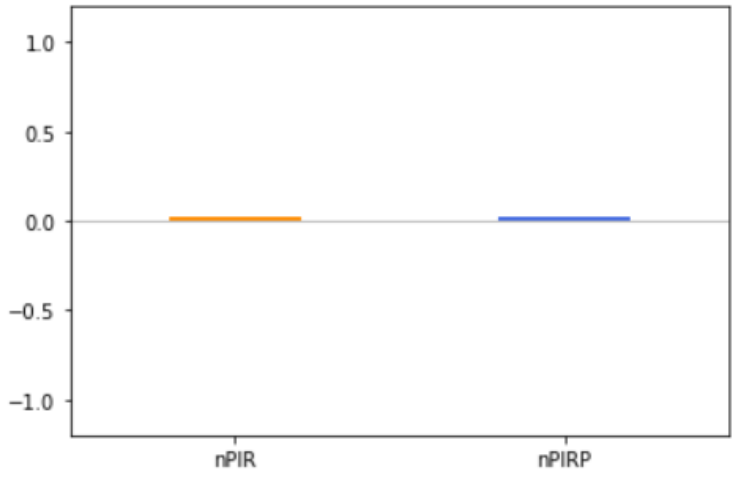

Figure 4.7. Example 3: Adverbs substitution perturbation

Sentence feature extraction and removal perturbation

Now it is applied the *removal perturbation* over the *sentence feature extraction* method.

In this simple example there are only two sentences and both are not influential for the prediction as showed in figures 4.8 and 4.9.

This behaviour of the model is how is expected because both sentences are very negative and removing only one of them at a time cannot change the prediction.

One aspect that emerges here is that different *feature extraction* methods look for different aspects of the input text and then to provides different explanations. This implies that all methods can be used in a complementary manner.

```
Removed ID-Token: [(0, 'This'), (1, 'film'), (2, 'was'),
(3, 'very'), (4, 'awful'), (5, '.')]

Description:  sentences_perturbation 1


Text With Perturbation:
This film was very awful. I have never seen such a bad movie.

Text After Perturbation:
I have never seen such a bad movie.


Probabilities: [0.9992480874061584, 0.0007519137579947 71]
Predicted Label after Perturbation: Negative


Predicted Label Before Perturbation: Negative
Original Label: Negative


nPIR: 0.0
nPIRP: -1.0
```

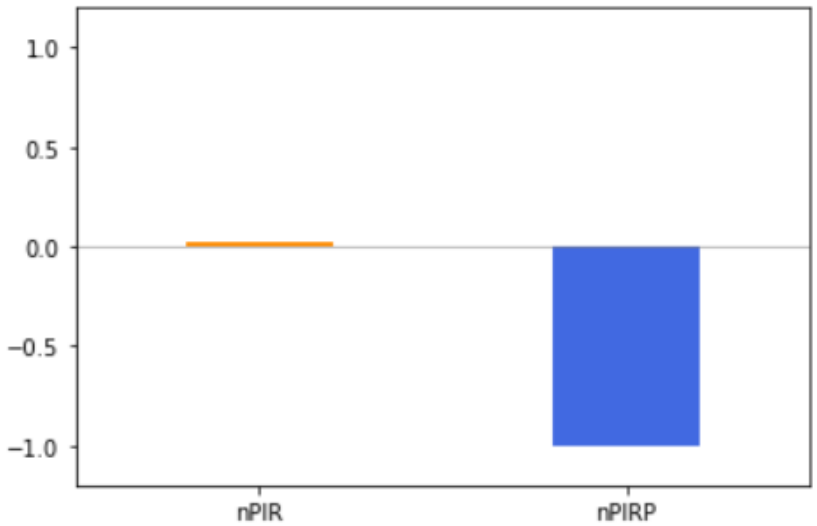

Figure 4.8. Example 3: Sentence 1 removal perturbation

```
Removed ID-Token: [(6, 'I'), (7, 'have'), (8, 'never'), (9, 'seen'),
(10, 'such'), (11, 'a'), (12, 'bad'), (13, 'movie'), (14, '.')]

Description:  sentences_perturbation 2

Text With Perturbation:
This film was very awful. I have never seen such a bad movie.

Text After Perturbation:
This film was very awful.

Probabilities: [0.9992656111717224, 0.0007343674078583717]
Predicted Label after Perturbation: Negative

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: 0.0
nPIRP: -1.0
```

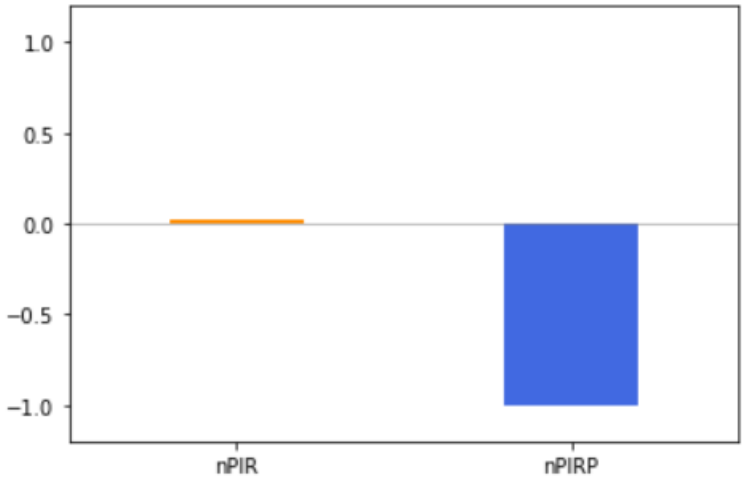

Figure 4.9. Example 3: Sentence 2 removal perturbation

Clustering feature extraction and removal perturbation

Finally is applied the *removal perturbation* over the *multi-layer word embedding feature extraction* method.

In this case the algorithm automatically discovers that the *top-k* is four. Following are reported all the four clusters defined by the *k-means* algorithm over the *word embedding* extracted from the model.

Analyzing the results emerges that the only *cluster* responsable for the prediction is the number two as showed in figure 4.12. This cluster is composed by the words *"was"*, *"awful"*, *"bad"* and *"movie"* and applying a *removal perturbation* over this feature the predicted label changes from *negative* to positive *positive* with a very high *nPIR* index. All the other clusters instead are not influential for the prediction as showed in figures 4.10, 4.11 and 4.13.

This behaviour of the model with this feature selection is how it is expected and the influential feature founded is very similar to the one founded by the part-of-speech. Moreover the clustering is able to group also words belonging to different *sentences* and to different *parts-of-speech*. In this case combines to adjectives like *bad* and *afwul* and closely related words like *was* and *film*.

This is the last part of the *local explanation* and, combined with the other explanations provided, can help to figures out why the model makes a particular prediction given an input review.

Removed ID-Token: [(3, 'very'), (7, 'have'), (11, 'a')]

Description: cluster_perturation_K_4_cluster_0

Text With Perturbation:
this film was very awful i have never seen such a bad movie

Text After Perturbation:
this film was awful i never seen such bad movie

Probabilities: [0.9990229606628418, 0.0009770413162186742]
Predicted Label after Perturbation: Negative

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: 0.001
nPIRP: -1.0

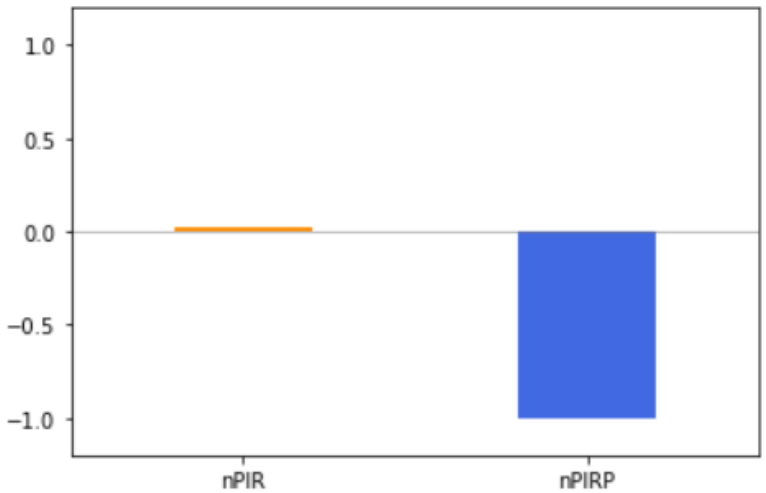

Figure 4.10. Example 3: Cluster 0 removal perturbation

Removed ID-Token: [(0, 'this'), (1, 'film'), (10, 'such')]

Description: cluster_perturation_K_4_cluster_1

Text With Perturbation:
this film was very awful i have never seen such a bad movie

Text After Perturbation:
was very awful i have never seen a bad movie

Probabilities: [0.998838484287262, 0.001161577645689249]
Predicted Label after Perturbation: Negative

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: 0.001
nPIRP: -1.0

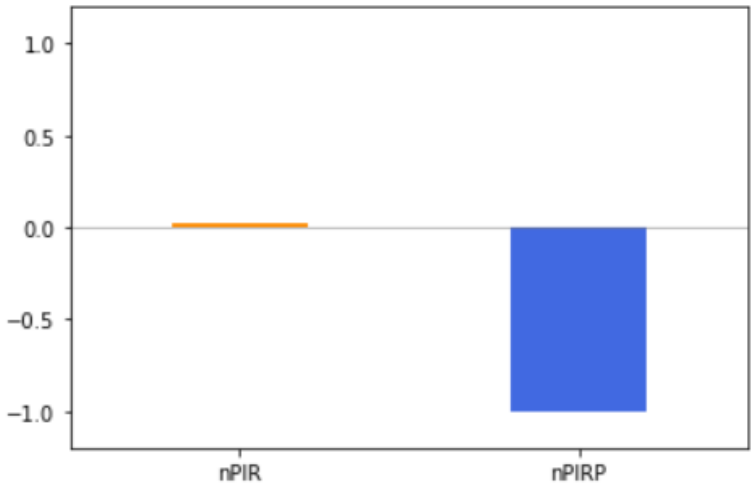

Figure 4.11. Example 3: Cluster 1 removal perturbation

```
Removed ID-Token: [(2, 'was'), (4, 'awful'), (12, 'bad'), (13, 'movie')]

Description:  cluster_perturation_K_4_cluster_2

Text With Perturbation:
this film was very awful i have never seen such a bad movie

Text After Perturbation:
this film very i have never seen such a

Probabilities: [0.016048947349190712, 0.9839510917663574]
Predicted Label after Perturbation: Positive

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: 0.984
nPIRP: -1.0
```

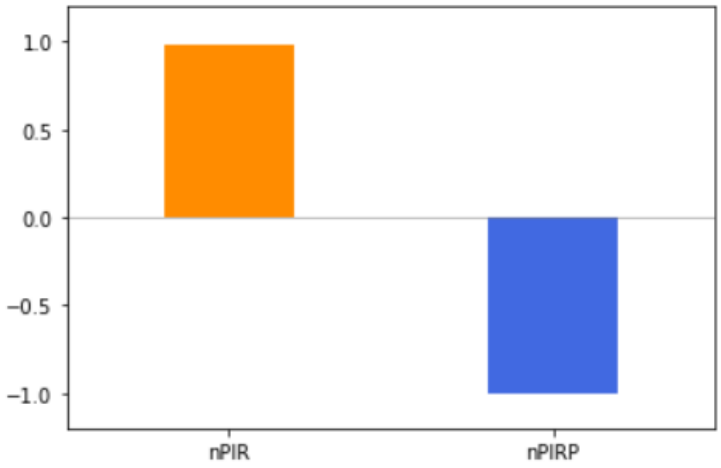

Figure 4.12. Example 3: Cluster 2 removal perturbation

Removed ID-Token: [(6, 'i'), (8, 'never'), (9, 'seen')]

Description: cluster_perturation_K_4_cluster_3

Text With Perturbation:
this film was very awful i have never seen such a bad movie

Text After Perturbation:
this film was very awful have such a bad movie

Probabilities: [0.9986868500709534, 0.0013131941668689 25]
Predicted Label after Perturbation: Negative

Predicted Label Before Perturbation: Negative
Original Label: Negative

nPIR: 0.001
nPIRP: -1.0

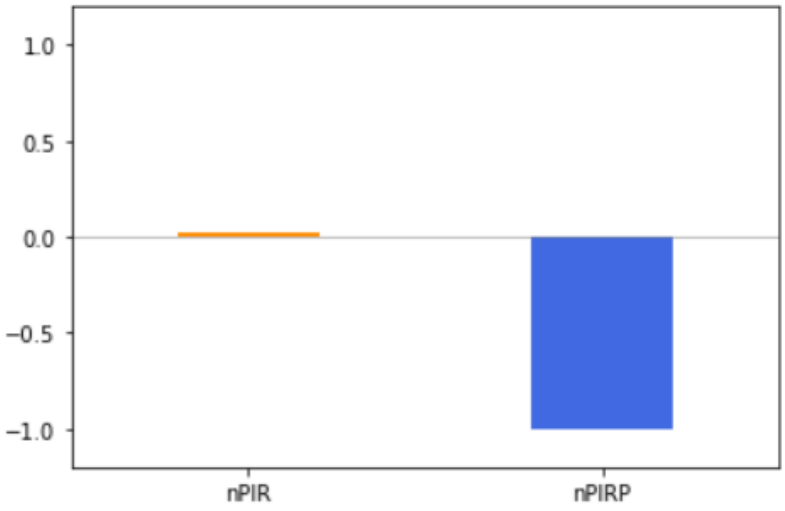

Figure 4.13. Example 3: Cluster 3 removal perturbation

4.3.2 Example of wrong prediction

In this example the model makes a wrong prediction with the original input text. A local explanation, due to the fact that the original predicted label was wrong, can help to figure out an answer to the question:
Why the model makes a wrong prediction for this input review?

Original prediction

Given the original input text, the model predicts that the review is negative with a percentage of 99% but the original label of the review is positive as showed in the figure 4.14. This implies that the model makes a wrong prediction on the original text and then the explanation can help to figure out why it happened.

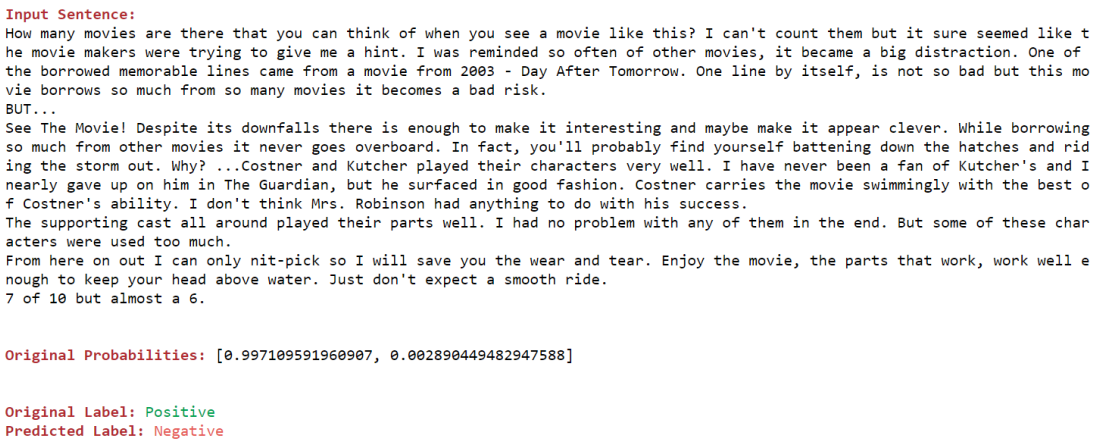

Figure 4.14. Example 2: original prediction

Part-of-speech feature extraction and removal perturbation

Starting from the *part-of-speech features extraction* method it is applied a *removal perturbation* over each selected words.

In this case the *adjectives* are influential for the prediction as showed in figures 4.15 and 4.16. After the perturbation over the adjectives, and then after removing each adjective, the predicted label changes. Due to the fact that in this example the original prediction of the model was wrong, emerges that the adjectives are one of the responsible of the wrong prediction.

Applying instead a *removal perturbation* over the *nouns* the predicted label not changes and this implies that *nouns* are not influential for the original prediction, as showed in figures 4.17 and 4.18.

Here are reported only these two examples of perturbation, what emerges by analyzing the full report is that only *adjectives* and *verbs* with a *removal perturbation* are positively influential, while all the other are not influential.

Removed ID-Token: [(1, 'many'), (45, 'other'), (51, 'big'), (58, 'memorable'), (79, 'bad'), (85, 'much'), (88, 'many'), (93, 'bad'), (107, 'enough'), (111, 'interesting'), (124, 'other'), (185, 'good'), (195, 'best'), (245, 'much'), (289, 'smooth')]

Description: Adjective perturbation

Text With Perturbation:
How many movies are there that you can think of when you see a movie like this? I ca n't count them but it sure seemed like the movie makers were trying to give me a hint. I was reminded so often of other movies, it became a big distraction. One of the borrowed memorable lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so bad but this movie borrows so much from so many movies it becomes a bad risk. BUT ... See The Movie! Despite its downfalls there is enough to make it interesting and maybe make it appear clever. While borrowing so much from other movies it never goes overboard. In fact, you 'll probably find yourself battening down the hatches and riding the storm out. Why? ... Costner and Kutcher played their characters very well. I have never been a fan of Kutcher 's and I nearly gave up on him in The Guardian, but he surfaced in good fashion. Costner carries the movie swimmingly with the best of Costner 's ability. I do n't think Mrs. Robinson had anything to do with his success. The supporting cast all around played their parts well. I had no problem with any of them in the end. But some of these characters were used too much. From here on out I can only nit-pick so I will save you the wear and tear. Enjoy the movie, the parts that work, work well enough to keep your head above water. Just do n't expect a smooth ride. 7 of 10 but almost a 6.

Text After Perturbation:
How movies are there that you can think of when you see a movie like this? I ca n't count them but it sure seemed like the movie makers were trying to give me a hint. I was reminded so often of movies, it became a distraction. One of the borrowed lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so but this movie borrows so from so movies it becomes a risk. BUT... See The Movie! Despite its downfalls there is to make it and maybe make it appear clever. While borrowing so much from movies it never goes overboard. In fact, you 'll probably find yourself battening down the hatches and riding the storm out. Why?... Costner and Kutcher played their characters very well. I have never been a fan of Kutcher 's and I nearly gave up on him in The Guardian, but he surfaced in fashion. Costner carries the movie swimmingly with the of Costner 's ability. I do n't think Mrs. Robinson had anything to do with his success. The supporting cast all around played their parts well. I had no problem with any of them in the end. But some of these characters were used too. From here on out I can only nit-pick so I will save you the wear and tear. Enjoy the movie, the parts that work, work well enough to keep your head above water. Just do n't expect a ride. 7 of 10 but almost a 6.

Figure 4.15. Example 2: Adjectives removal perturbation

```
Probabilities: [0.1163620576262474, 0.8836379051208496]
Predicted Label after Perturbation: Positive

Predicted Label Before Perturbation: Negative
Original Label: Positive

nPIR: 0.884
nPIRP: -1.0
```

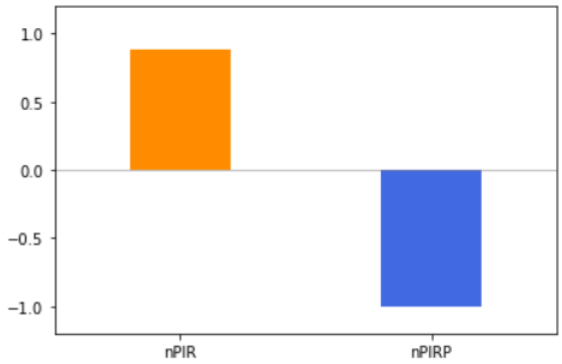

Figure 4.16. Example 2: Adjectives removal perturbation scores

```
Removed ID-Token: [(2, 'movies'), (14, 'movie'), (29, 'movie'), (30, 'makers'), (37, 'hint'), (46, 'movies'), (52, 'distract
ion'), (59, 'lines'), (63, 'movie'), (67, 'Day'), (69, 'Tomorrow'), (72, 'line'), (82, 'movie'), (89, 'movies'), (94, 'ris
k'), (96, 'BUT'), (100, 'Movie'), (104, 'downfalls'), (117, 'clever'), (125, 'movies'), (132, 'fact'), (142, 'hatches'), (14
6, 'storm'), (152, 'Costner'), (154, 'Kutcher'), (157, 'characters'), (166, 'fan'), (168, 'Kutcher'), (179, 'Guardian'), (18
6, 'fashion'), (188, 'Costner'), (191, 'movie'), (197, 'Costner'), (199, 'ability'), (205, 'Mrs.'), (206, 'Robinson'), (208,
'anything'), (213, 'success'), (217, 'cast'), (222, 'parts'), (228, 'problem'), (235, 'end'), (241, 'characters'), (261, 'we
ar'), (263, 'tear'), (267, 'movie'), (270, 'parts'), (280, 'head'), (282, 'water'), (284, 'Just'), (290, 'ride')]

Description:  Noun perturbation

Text With Perturbation:
How many movies are there that you can think of when you see a movie like this? I ca n't count them but it sure seemed like
the movie makers were trying to give me a hint. I was reminded so often of other movies, it became a big distraction. One of
the borrowed memorable lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so bad but this mo
vie borrows so much from so many movies it becomes a bad risk. BUT ... See The Movie! Despite its downfalls there is enough
to make it interesting and maybe make it appear clever. While borrowing so much from other movies it never goes overboard. I
n fact, you 'll probably find yourself battening down the hatches and riding the storm out. Why? ... Costner and Kutcher pla
yed their characters very well. I have never been a fan of Kutcher 's and I nearly gave up on him in The Guardian, but he su
rfaced in good fashion. Costner carries the movie swimmingly with the best of Costner 's ability. I do n't think Mrs. Robins
on had anything to do with his success. The supporting cast all around played their parts well. I had no problem with any of
them in the end. But some of these characters were used too much. From here on out I can only nit-pick so I will save you th
e wear and tear. Enjoy the movie, the parts that work, work well enough to keep your head above water. Just do n't expect a
smooth ride. 7 of 10 but almost a 6.

Text After Perturbation:
How many are there that you can think of when you see a like this? I ca n't count them but it sure seemed like the were tryi
ng to give me a. I was reminded so often of other, it became a big. One of the borrowed memorable came from a from 2003 - Af
ter. One by itself, is not so bad but this borrows so much from so many it becomes a bad.... See The! Despite its there is e
nough to make it interesting and maybe make it appear. While borrowing so much from other it never goes overboard. In, you
'll probably find yourself battening down the and riding the out. Why?... and played their very well. I have never been a of
's and I nearly gave up on him in The, but he surfaced in good. carries the swimmingly with the best of 's. I do n't think h
ad to do with his. The supporting all around played their well. I had no with any of them in the. But some of these were use
d too much. From here on out I can only nit-pick so I will save you the and. Enjoy the, the that work, work well enough to k
eep your above. do n't expect a smooth. 7 of 10 but almost a 6.
```

Figure 4.17. Example 2: Nouns removal perturbation

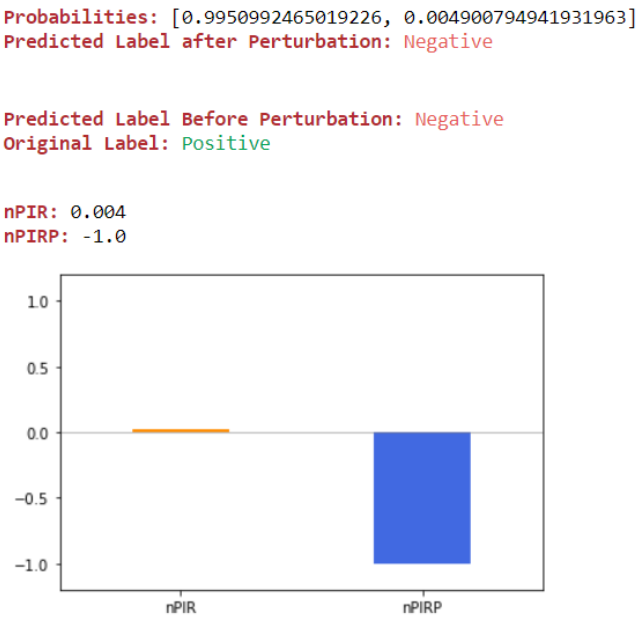

Figure 4.18. Example 2: Nouns removal perturbation scores

Part-of-speech feature extraction and substitution perturbation

This time, starting from the *part-of-speech feature extraction*, is applied a *substitution perturbation*.

In this case *adjectives* are *substituted* with antonyms and what emerges is that this type of perturbation not changes the predicted label. This is interesting because the model predicts the same label both with original adjectives and their antonyms. This behaviour is incoherent with the human reasoning because the model originally did wrong, then the antonyms are captured correctly and then, comparing with the removal perturbation of the same adjectives, emerges that the original adjectives are not understood correctly in this context while their antonyms are understood correctly. This is because, also for humans, is reasonable that, if the original label was positive, changing the adjectives with their antonyms the label becomes negative.

The *adjective feature extraction* with *substitution perturbation* is showed in figures 4.19 and 4.20.

Also in this case is reported only one example for simplicity but analyzing the full report emerges that there are not part-of-speech with the substitution perturbation that change the predicted label.

Substituted ID-Token-Antonym: [(1, 'many', 'few'), (45, 'other', 'same'), (51, 'big', 'little'), (79, 'bad', 'good'), (85, 'much', 'little'), (88, 'many', 'few'), (93, 'bad', 'good'), (111, 'interesting', 'bore'), (124, 'other', 'same'), (185, 'good', 'evil'), (195, 'best', 'worst'), (245, 'much', 'little'), (289, 'smooth', 'roughen')]

Description: Adjective substitution with antonyms perturbation

Text With Perturbation:
How ("many") few movies are there that you can think of when you see a movie like this? I ca n't count them but it sure seemed like the movie makers were trying to give me a hint. I was reminded so often of ("other") same movies, it became a ("big") little distraction. One of the borrowed memorable lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so ("bad") good but this movie borrows so ("much") little from so ("many") few movies it becomes a ("bad") good risk. BUT ... See The Movie! Despite its downfalls there is enough to make it ("interesting") bore and maybe make it appear clever. While borrowing so much from ("other") same movies it never goes overboard. In fact, you 'll probably find yourself battening down the hatches and riding the storm out. Why? ... Costner and Kutcher played their characters very well. I have never been a fan of Kutcher 's and I nearly gave up on him in The Guardian, but he surfaced in ("good") evil fashion. Costner carries the movie swimmingly with the ("best") worst of Costner 's ability. I do n't think Mrs. Robinson had anything to do with his success. The supporting cast all around played their parts well. I had no problem with any of them in the end. But some of these characters were used too ("much") little. From here on out I can only nit-pick so I will save you the wear and tear. Enjoy the movie, the parts that work, work well enough to keep your head above water. Just do n't expect a ("smooth") roughen ride. 7 of 10 but almost a 6.

Text After Perturbation:
How few movies are there that you can think of when you see a movie like this? I ca n't count them but it sure seemed like the movie makers were trying to give me a hint. I was reminded so often of same movies, it became a little distraction. One of the borrowed memorable lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so good but this movie borrows so little from so few movies it becomes a good risk. BUT... See The Movie! Despite its downfalls there is enough to make it bore and maybe make it appear clever. While borrowing so much from same movies it never goes overboard. In fact, you 'll probably find yourself battening down the hatches and riding the storm out. Why?... Costner and Kutcher played their characters very well. I have never been a fan of Kutcher 's and I nearly gave up on him in The Guardian, but he surfaced in evil fashion. Costner carries the movie swimmingly with the worst of Costner 's ability. I do n't think Mrs. Robinson had anything to do with his success. The supporting cast all around played their parts well. I had no problem with any of them in the end. But some of these characters were used too little. From here on out I can only nit-pick so I will save you the wear and tear. Enjoy the movie, the parts that work, work well enough to keep your head above water. Just do n't expect a roughen ride. 7 of 10 but almost a 6.

Figure 4.19. Example 2: Adjectives substitution perturbation

Probabilities: [0.9846883416175842, 0.015311659313738346]
Predicted Label after Perturbation: Negative

Predicted Label Before Perturbation: Negative
Original Label: Positive

nPIR: 0.024
nPIRP: -1.0

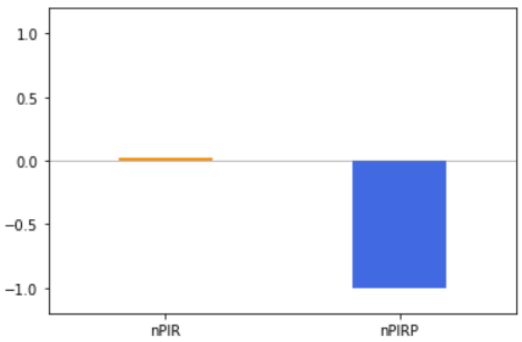

Figure 4.20. Example 2: Adjectives substitution perturbation scores

Sentence feature extraction and removal perturbation

Now are reported some examples of *sentence feature extraction* with a *removal perturbation*.

The sentence showed in figures 4.21 and 4.22 is an example of perturbation that changes the class label. This sentence then is one of the responsible of the wrong prediction.

The figures 4.23 and 4.24 show an example of a non influential perturbation that not changes the class label.

Also in this case are reported two examples for simplicity and what emerges looking the full report is that only the third sentence changes the class label while all the other are not influential and then not change the predicted label.

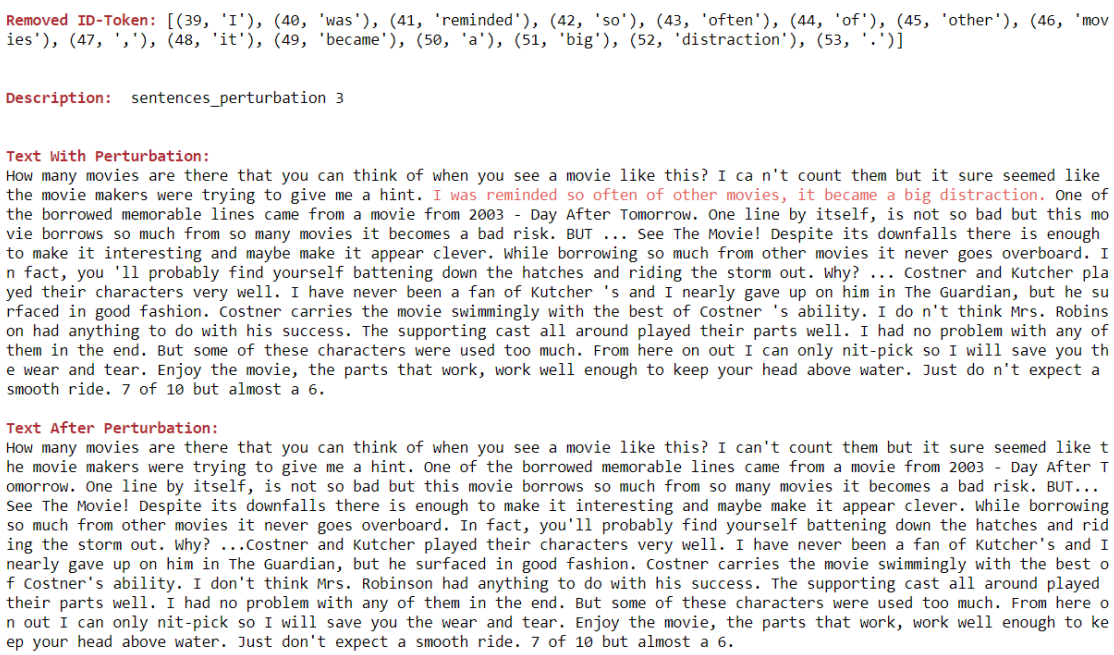

Figure 4.21. Example 2: Sentence 3 removal perturbation

Probabilities: [0.36408716440200806, 0.6359127759933472]
Predicted Label after Perturbation: Positive

Predicted Label Before Perturbation: Negative
Original Label: Positive

nPIR: 0.663
nPIRP: -1.0

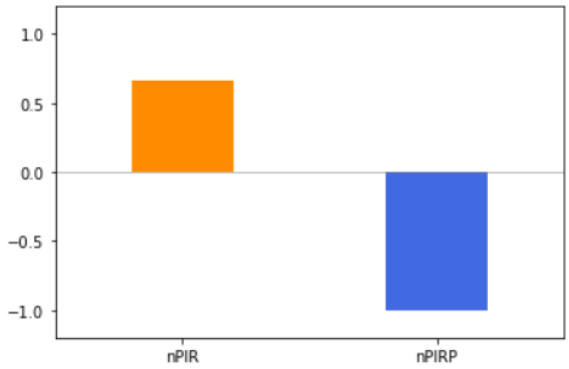

Figure 4.22. Example 2: Sentence 3 removal perturbation scores

Removed ID-Token: [(102, 'Despite'), (103, 'its'), (104, 'downfalls'), (105, 'there'), (106, 'is'), (107, 'enough'), (108, 'to'), (109, 'make'), (110, 'it'), (111, 'interesting'), (112, 'and'), (113, 'maybe'), (114, 'make'), (115, 'it'), (116, 'appear'), (117, 'clever'), (118, '.')]

Description: sentences_perturbation 7

Text With Perturbation:
How many movies are there that you can think of when you see a movie like this? I ca n't count them but it sure seemed like the movie makers were trying to give me a hint. I was reminded so often of other movies, it became a big distraction. One of the borrowed memorable lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so bad but this movie borrows so much from so many movies it becomes a bad risk. BUT ... See The Movie! Despite its downfalls there is enough to make it interesting and maybe make it appear clever. While borrowing so much from other movies it never goes overboard. In fact, you 'll probably find yourself battening down the hatches and riding the storm out. Why? ... Costner and Kutcher played their characters very well. I have never been a fan of Kutcher 's and I nearly gave up on him in The Guardian, but he surfaced in good fashion. Costner carries the movie swimmingly with the best of Costner 's ability. I do n't think Mrs. Robinson had anything to do with his success. The supporting cast all around played their parts well. I had no problem with any of them in the end. But some of these characters were used too much. From here on out I can only nit-pick so I will save you the wear and tear. Enjoy the movie, the parts that work, work well enough to keep your head above water. Just do n't expect a smooth ride. 7 of 10 but almost a 6.

Text After Perturbation:
How many movies are there that you can think of when you see a movie like this? I can't count them but it sure seemed like the movie makers were trying to give me a hint. I was reminded so often of other movies, it became a big distraction. One of the borrowed memorable lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so bad but this movie borrows so much from so many movies it becomes a bad risk. BUT...
See The Movie! While borrowing so much from other movies it never goes overboard. In fact, you'll probably find yourself battening down the hatches and riding the storm out. Why? ...Costner and Kutcher played their characters very well. I have never been a fan of Kutcher's and I nearly gave up on him in The Guardian, but he surfaced in good fashion. Costner carries the movie swimmingly with the best of Costner's ability. I don't think Mrs. Robinson had anything to do with his success. The supporting cast all around played their parts well. I had no problem with any of them in the end. But some of these characters were used too much. From here on out I can only nit-pick so I will save you the wear and tear. Enjoy the movie, the parts that work, work well enough to keep your head above water. Just don't expect a smooth ride. 7 of 10 but almost a 6.

Figure 4.23. Example 2: Sentence 7 removal perturbation

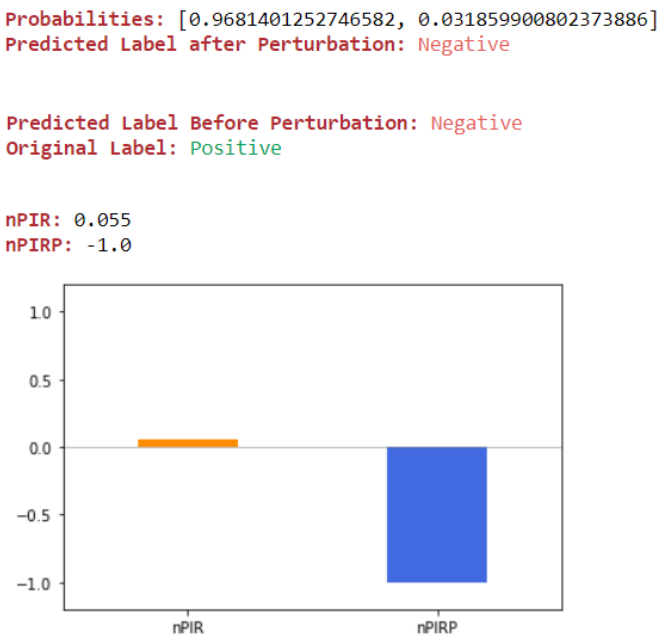

Figure 4.24. Example 2: Sentence 7 removal perturbation scores

Clustering features selection and removal perturbation

In the last part of this example is showed the *multi-layer word embedding feature extraction* to which is applied a *removal perturbation*.

The seventh cluster founded by the algorithm is positively influential for the prediction as showed in figures 4.25 and 4.26, while the ninth cluster is not influential as showed in figures 4.27 and 4.28. Also in this case are reported only two examples but in the seventh cluster is the only one responsable for the prediction by using the clustering algorithm.

In this case the algorithm has automatically figured out that the *top-k* is 15. One interesting aspect captured by this example is that the *multi-layer word embedding feature extraction* favors smaller groups that change more the prediction for how it is implemented because it weights the score for the number of words inside the feature in order to understand which is the *top-k*. It can be very interesting to analyze the behaviour of the model in this case where, in the first case are removed only two words and the prediction changes, while in the second case are removed a lot of words but the predicted class label not changes.

Removed ID-Token: [(4, 'there'), (108, 'there')]

Description: cluster_perturation_K_15_cluster_7

Text With Perturbation:
how many movies are there that you can think of when you see a movie like this i can t count them but it sure seemed like th
e movie makers were trying to give me a hint i was reminded so often of other movies it became a big distraction one of the
borrowed memorable lines came from a movie from 2003 day after tomorrow one line by itself is not so bad but this movie borr
ows so much from so many movies it becomes a bad risk but see the movie despite its downfalls there is enough to make it int
eresting and maybe make it appear clever while borrowing so much from other movies it never goes overboard in fact you ll pr
obably find yourself battening down the hatches and riding the storm out why costner and kutcher played their characters ver
y well i have never been a fan of kutcher s and i nearly gave up on him in the guardian but he surfaced in good fashion cost
ner carries the movie swimmingly with the best of costner s ability i don t think mrs robinson had anything to do with his s
uccess the supporting cast all around played their parts well i had no problem with any of them in the end but some of these
characters were used too much from here on out i can only nit pick so i will save you the wear and tear enjoy the movie the
parts that work work well enough to keep your head above water just don t expect a smooth ride 7 of 10 but almost a 6

Text After Perturbation:
how many movies are that you can think of when you see a movie like this i can t count them but it sure seemed like the movi
e makers were trying to give me a hint i was reminded so often of other movies it became a big distraction one of the borrow
ed memorable lines came from a movie from 2003 day after tomorrow one line by itself is not so bad but this movie borrows so
much from so many movies it becomes a bad risk but see the movie despite its downfalls is enough to make it interesting and
maybe make it appear clever while borrowing so much from other movies it never goes overboard in fact you ll probably find y
ourself battening down the hatches and riding the storm out why costner and kutcher played their characters very well i have
never been a fan of kutcher s and i nearly gave up on him in the guardian but he surfaced in good fashion costner carries th
e movie swimmingly with the best of costner s ability i don t think mrs robinson had anything to do with his success the sup
porting cast all around played their parts well i had no problem with any of them in the end but some of these characters we
re used too much from here on out i can only nit pick so i will save you the wear and tear enjoy the movie the parts that wo
rk work well enough to keep your head above water just don t expect a smooth ride 7 of 10 but almost a 6

Figure 4.25. Example 2: Cluster 7 removal perturbation

Probabilities: [0.37783652544021606, 0.6221634745597839]
Predicted Label after Perturbation: Positive

Predicted Label Before Perturbation: Negative
Original Label: Positive

nPIR: 0.651
nPIRP: -1.0

Figure 4.26. Example 2: Cluster 7 removal perturbation scores

Removed ID-Token: [(6, 'you'), (10, 'when'), (12, 'see'), (22, 'count'), (24, 'but'), (26, 'sure'), (28, 'like'), (31, 'makers'), (33, 'trying'), (34, 'to'), (36, 'me'), (38, 'hint'), (42, 'reminded'), (52, 'big'), (58, 'borrowed'), (59, 'memorable'), (61, 'came'), (68, 'day'), (69, 'after'), (70, 'tomorrow'), (72, 'one'), (74, 'by'), (78, 'not'), (81, 'but'), (84, 'borrows'), (92, 'becomes'), (94, 'bad'), (97, 'but'), (105, 'despite'), (107, 'downfalls'), (110, 'enough'), (111, 'to'), (114, 'interesting'), (120, 'clever'), (122, 'while'), (123, 'borrowing'), (132, 'overboard'), (137, 'you'), (139, 'll'), (140, 'probably'), (143, 'battening'), (144, 'down'), (146, 'hatches'), (150, 'storm'), (151, 'out'), (153, 'why'), (160, 'kutcher'), (161, 'played'), (162, 'their'), (163, 'characters'), (164, 'very'), (165, 'well'), (168, 'have'), (169, 'never'), (170, 'been'), (174, 'kutcher'), (181, 'up'), (182, 'on'), (183, 'him'), (184, 'in'), (186, 'guardian'), (189, 'he'), (192, 'good'), (193, 'fashion'), (195, 'costner'), (199, 'swimmingly'), (204, 'costner'), (210, 'don'), (213, 'think'), (214, 'mrs'), (219, 'to'), (220, 'do'), (226, 'supporting'), (228, 'all'), (232, 'parts'), (237, 'no'), (238, 'problem'), (240, 'any'), (243, 'in'), (245, 'end'), (247, 'but'), (248, 'some'), (251, 'characters'), (253, 'used'), (258, 'here'), (259, 'on'), (260, 'out'), (263, 'only'), (264, 'nit'), (266, 'pick'), (269, 'will'), (271, 'you'), (273, 'wear'), (275, 'tear'), (277, 'enjoy'), (282, 'parts'), (283, 'that'), (284, 'work'), (286, 'work'), (287, 'well'), (288, 'enough'), (289, 'to'), (290, 'keep'), (291, 'your'), (293, 'above'), (294, 'water'), (296, 'just'), (297, 'don'), (300, 'expect'), (302, 'smooth'), (305, '7'), (308, 'but'), (311, '6')]

Description: cluster_perturation_K_15_cluster_9

Text With Perturbation:
how many movies are there that you can think of when you see a movie like this i can t count them but it sure seemed like the movie makers were trying to give me a hint i was reminded so often of other movies it became a big distraction one of the borrowed memorable lines came from a movie from 2003 day after tomorrow one line by itself is not so bad but this movie borrows so much from so many movies it becomes a bad risk but see the movie despite its downfalls there is enough to make it interesting and maybe make it appear clever while borrowing so much from other movies it never goes overboard in fact you ll probably find yourself battening down the hatches and riding the storm out why costner and kutcher played their characters very well i have never been a fan of kutcher s and i nearly gave up on him in the guardian but he surfaced in good fashion costner carries the movie swimmingly with the best of costner s ability i don t think mrs robinson had anything to do with his success the supporting cast all around played their parts well i had no problem with any of them in the end but some of these characters were used too much from here on out i can only nit pick so i will save you the wear and tear enjoy the movie the parts that work work well enough to keep your head above water just don t expect a smooth ride 7 of 10 but almost a 6

Text After Perturbation:
how many movies are there that can think of you a movie like this i can t them it seemed the movie were give a i was so often of other movies it became a distraction one of the lines from a movie from 2003 line itself is so bad this movie so much from so many movies it a risk see the movie its there is make it and maybe make it appear so much from other movies it never goes in fact find yourself the and riding the costner and i a fan of s and i nearly gave the but surfaced in carries the movie with the best of s ability i t robinson had anything with his success the cast around played their well i had with of the m the of these were too much from i can so i save the and the movie the head t a ride of 10 almost a

Figure 4.27. Example 2: Cluster 9 removal perturbation

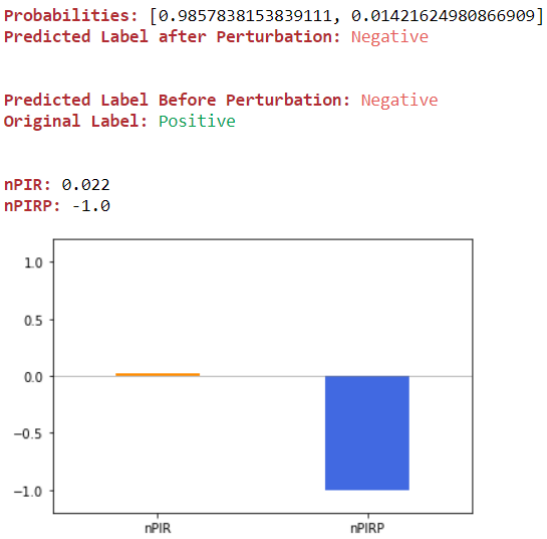

Figure 4.28. Example 2: Cluster 9 removal perturbation scores

4.4 Global results

In the previous sections are analyzed the local results of the methodology over a single input example at a time. One further analysis can be done globally for more input texts together. To this purpose has been create a dataset of different transparency reports, each one with the local explanation of a different input text. The input texts analyzed are taken from the test set of the IMDB dataset randomly. Each of this trasparency report contains all the perturbations for each type of features selection.

4.4.1 Global statistics

With the purpose to measure how many times EBANO Text can find a influential feature selection and also to make a comparison between the different feature extraction techniques it is done the following analysis.

For each different feature extraction is calculated, in percentage, how many time finds at least one influential perturbation (with nPIR ≥ 0.5). This calculation it is done separately for all different feature extraction type and also with all the three methods together. Beyond this it is also calculated the same percentage but taking also perturbation on two-by-two combination of the selected features. For example combining adjective and noun to create a bigger perturbation, or combine two different senteces or again combining two different clusters.

NB: In the following analysis are reported only the *Removal Perturbations*.

Feature Extraction Type	No Combinations	2 Combinations
Part-of-speech	35%	78%
Sentence FE	22%	30%
Word Embedding	78%	92%
All	82%	95%

Table 4.1. Global results.

In the table 4.1 are reported the percentage of time where each feature extraction method found at least one influential perturbation for each input text with respect to the total number of input texts.

As the table shows, the *Multi-layer Word Embedding feature extraction* has about the twice of probability of found a influential removal perturbation with respect of the other two methods if two-by-two combinations of them are not taken into account.

The *sentence feature selection*, without taking into account combinations, only in the 22% of cases finds a influential perturbation. The main problem of this feature extraction type is that when occurs in very long review it removes a small subset and then the model can still understand the context without change the prediction. With this type of feature selection the words analyzed are not function of the size of the review but are fixed to the sentence length. When attempt with small review sentence feature extraction works quite well, with medium-large review not find often a good feature selection, but when find it is very interesting and very understandable in human terms. Also the combination two-by-two of sentences do not increases much the percentage because with large review also two sentences is a very small subset of the entirely text and have a percentage of 30%. In the dataset are present review with also 4000 words and then about 40 sentences.

As regard instead for *part-of-speech feature selection* again it has a low percentage of find a influential feture selection without considering combinations that is 35%, in this case the number of words taken into account is function of the size of the review in that larger review will have larger set of adjectives,nouns,verbs and so on. In this case what emerges is that by taking into account also the two-by-two combinations of different parts-of-speech like for example combining adjectives and nouns or adjectives and verbs and so on the previous percentage becomes more then twice with a value of 75%. This is due to the fact that also for humans is very different removing only the word *"good"* or only the word *"film"* like in a single part-of-speech feature selection, or removing the combination of *"good film"*.

Instead *multi-layer word embedding feature selection* for its nature can take into account words that are in different sentences, but also words inside the same sentence but belonging to different parts-of-speech and then solves the problems of the others two cases. For this reason without the combination of clusters the percentage is relative high with a 82% and by taking into account also the two-by-two combination of this similar cluster of words the percentage grown until 95%. Moreover the clustering algorithm has some

randomness inside and by multiple running of the same algorithm this percentage grown again. Furthermore the accuracy of the model, and then also its inner representation of word embedding, is under the 90% and probably for this reason the accuracy of word embedding presents some errors.

Another quantity of error is added by the type of perturbation, indeed the *removal perturbation* applied on texts is not the same to the one applied on images. Applied a removal of a portion of image in the input leads on removing a concept and in the context of object detection works very well, in text instead words are linked and is more complicated because the underlying sentiment can remains good. Probably the removal perturbation for another type of task like topic detection can have also better results.

Let clarify with an example:

- Image classification: if have a image classification algorithm that detects cats, and you remove the specific portion of the image with the cat it is expected that the algorithm not recognize again the perturbed image as a cat but the label not cat because the perturbation implies a full removal of the concept.

- Text classification:

    1. Sentiment classification: if have for example the sentence "this is a good film" and removing the feature "good" the new perturbed text becomes "this is a film". For human this sentence is neutral than it is expected that the probability to be positive is lower but not that is negative this cause some imprecision.
    2. Topic detection: in topic detection task instead is more likely that the removal of some words will implies to remove the concept. For example a text that speaking about sport, if removing this words it is expected that the prediction is not sport and the removal will be more precise.

For this consideration it was introduced the *substitution perturbation* where, instead of simple removing one concept, it is changed with a possible opposite concept. Returning on the previous example, also for humans, is more reasonable that removing "good" and substituting it with the word "bad" the sentiment become negative and then this is the concept that is influential for the prediction.

4.4.2 Frequent influential words

For the class label *Positive* and then all the input texts that are originally predicted as positive reviews, are taken all the more influential *Removal Perturbation* (with nPIR ≥ 0.9) for all the different types of *Interpretable Features*. By analyzing all the words that belong this features it is created a *Word Cloud* with the frequency occurrences of them as showed in figure 4.29.

From the picture emerges that in very positive features appear nouns that are sentiment neutral like "movie", "film", "story" that are responsible of express the concept and some adjectives like "great", "good" and so on.
In the picture appears also the word "weapons", maybe the reason can be that the model was trained with a lot of positive reviews of war films and then the model associates the weapons to a positive sentiment.

Figure 4.29. Wordcloud Positive class label

The same analysis is done for the class label *Negative* as showed in the figure 4.30.

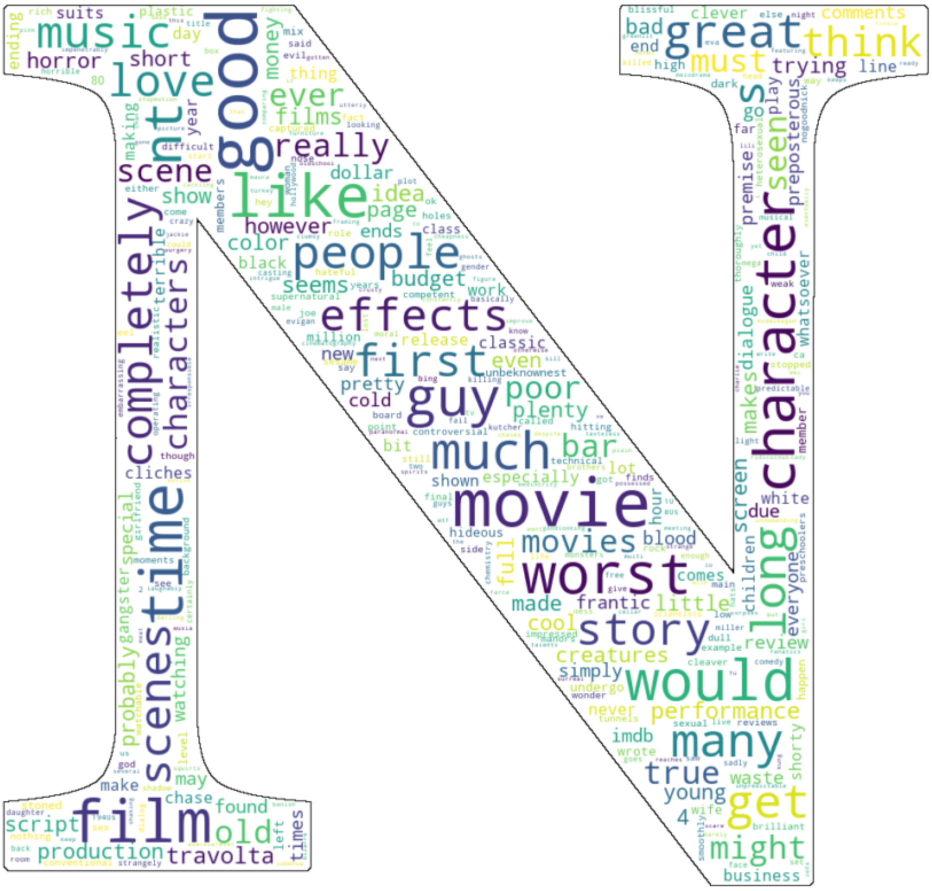

Figure 4.30. Wordcloud Negative class label

And also for both *Positive* and *Negative* labels looking which are in general the more influential words as showed in figure 4.31.

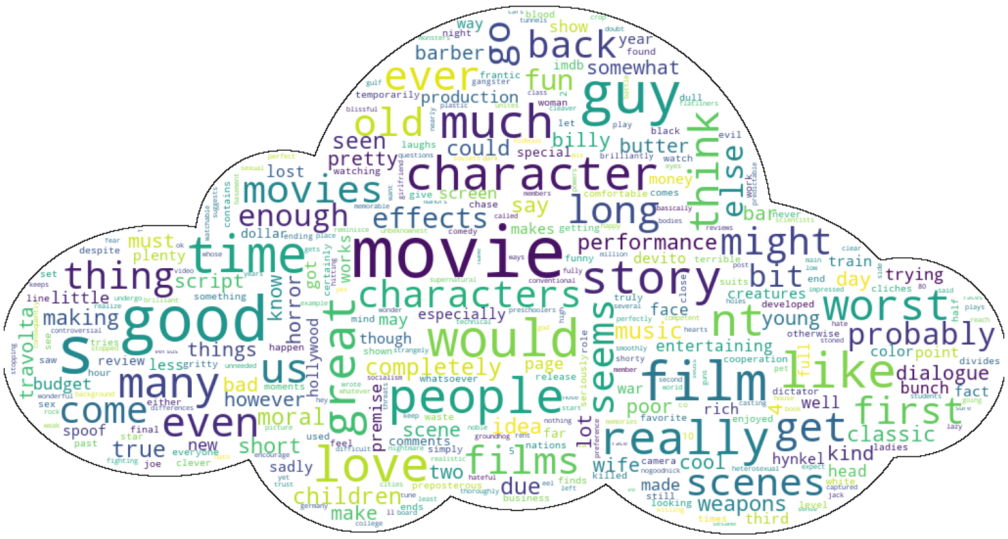

Figure 4.31. Wordcloud most influential words

Chapter 5

Conclusion and Future Work

In this final chapter are presented the final conclusions of this thesis showing some considerations of the different feature extraction and perturbation methods experimented. This part attempts to highlight the more innovative aspects of the proposed methodology and the results obtained. Besides this are proposed some possible improvements and future works to make the methodology more general, flexible and effective.

5.1 Conclusion

As explained in the previous chapters the objective of this thesis is to propose and experiment a methodology for the explanation of black-box models in the context of natural language processing. The need for this work emerged after a study of state-of-the-art explanation's techniques that showed a lack in the natural language processing field.

The proposed methodology, called *EBANO Text*, consists of provide a *local explanation* with a process of perturbation over a set of extracted features in order to measure the impact of the influence of each feature over the original predicted output.

The different methods of feature extraction proposed analyze different aspects of the input text. The *part-of-speech feature extraction* figure out if there is a particular part-of-speech that has influenced the prediction. It is human understandable because provides as explanation which is the part-of-speech influential for the prediction and the words that belong to it. This method become very powerful by taking into account also the two-by-two combination by reaching an high accuracy. The *sentence feature extraction* method instead figure out if there is a particular sentence influential for the prediction. This method is probably the most human understandable one because provides as explanation a whole sentence and the words the belong to it, but unfortunately is the method that found an influential feature with the lower frequency. Finally, the *multi-layer word embedding feature extraction* method is the most interesting and innovative one because look the internal learned feature of the model, contextualized in the input text, to figure out which are the words influential for the prediction. It provides as output explanation only the words influential and their positions, instead of the part-of-speech or the whole sentence, and reaches an high accuracy both with or without two-by-two combinations. One of the aspects that differentiates this approach from the state-of-the-art is that, taking into account different feature extraction methods, it can provides different types of explanation.

Another aspect that differentiates the proposed methodology from the others state-of-the-art techniques is that experiments more type of perturbation because, while the *removal perturbation* analyze the absence of a concept, the *substitution perturbation* is potentially more powerful because analyze the replacement of a concept with its opposite.

The different *feature extraction* methods and the different *perturbation* methods are complementary, because look different aspects of the input, and together can provide a better *local explanation*. The comparison between combinations of these techniques provides different results that can be very helpful to understand the behaviour of the model and then, consequently, reinforcing the explanation.

Another positive aspect of the proposed approach is that the *visual explanation* highlights the words in the input text showing the context very clearly and combined with the *numerical explanation*, given by the two indexes, measure in a simple and intuitive way how much each feature is important for the output prediction. These aspects are important to make the local explanation as human understandable as possible. The greatest complications encountered arises from the complex nature of human language, indeed each different combination of words provides a different possible result and this makes the analysis of results more difficult.

The methodology has been experimented on a large corpus of input texts and almost in all cases is founded at least once feature influential for the original prediction. Moreover the methodology has been experimented using *BERT* as deep neural network, that is a very advanced model which is very effective in understanding the context. This implied that the main objective has been reached providing in each case a *local explanation* that explain why the model, that is one of the most sophisticated available nowadays, makes a particular output prediction. Moreover, by analyzing a large corpus of input texts, it has been created a dataset of local explanations that allows to understand which are the most influential words for each class label. These most influential words are the ones learned as important by the model during the training process and this analysis allows to figure out if during the training errors or some forms of bias have occurred.

In conclusion, although there is still a lot of work to do, the hope is that this work can help the *eXplainable Artificial Intelligence* community by providing a framework to better understand the models that are used in the natural language processing field. Machine learning will probably continue to grow in the future and is important that also the explainable artificial intelligence will keep pace to provide models more trusted and understandable by humans.

5.2 Future works

In this work is experimented the binary text classification task, one possible evolution can be to explore the multi-class text classification field for example by fine-tuning the model for topic detection. To do this the architecture is already compliant, what changes mainly is the BERT's model fine-tuning.

Besides this, a possible improvement of the feature extraction methods, can be looking the attention values of the model to figure out the context of each word with a particular focus on the [CLS] token that is the responsable of the classification. Attention is a key aspect of the transformer encoder architecture of BERT and then can be very interesting analyze also this.

Other types of perturbation can be experimented, for example the insertion of new words randomly in the text, and the substitution perturbation can be improved in that the antonyms founded are not always coherent with the context.

Moreover another interesting improvement of the methodology can be, once founded a list of words belonging the same feature that are influential for the prediction, analyze how much each word impact the output to figure out which ones are the more influential. This could allows to provide a better visual and numerical explanation by highlights with different shades inside the different feature selection based on the numerical values.

Finally the methodology can be experimented with the larger version of BERT and maybe also different sequence models in the context of natural language processing. This aspect requires more computational power. While for the training phase the computational power is not a problem because it is done one time, for the feature extraction and the prediction phase can be a bottleneck.

Looking instead the user friendly feature of the framework can be very useful to provide an interactive front end to create and query the transparency reports both individually, to provide the local explanation, and jointly to provide a kind of global understanding of the model.

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. arXiv: 1409.0473 [cs.CL].

[2] Tolga Bolukbasi et al. "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings". In: *CoRR* abs/1607.06520 (2016). arXiv: 1607.06520. URL: http://arxiv.org/abs/1607.06520.

[3] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555.

[4] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[5] Riccardo Guidotti et al. "A Survey Of Methods For Explaining Black Box Models". In: *CoRR* abs/1802.01933 (2018). arXiv: 1802.01933. URL: http://arxiv.org/abs/1802.01933.

[6] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[7] Rani Horev. *BERT Explained: State of the art language model for NLP*. Ed. by Toward Data Science. URL: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[8] Edward Loper and Steven Bird. "NLTK: The Natural Language Toolkit". In: *CoRR* cs.CL/0205028 (2002). URL: http://dblp.uni-trier.de/db/journals/corr/corr0205.html#cs-CL-0205028.

[9] Andrew L. Maas et al. “Learning Word Vectors for Sentiment Analysis”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[10] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[11] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 3111–3119. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

[12] Andrew NG. *AI For Everyone*. Ed. by Coursera. URL: https://www.coursera.org/learn/ai-for-everyone/home/welcome.

[13] Andrew NG, Kian Katanforoosh, and Younes Bensouda Mourri. *Sequence Models*. Ed. by Coursera. URL: https://www.coursera.org/learn/nlp-sequence-models/home/welcome.

[14] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006.

[15] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[16] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “Glove: Global vectors for word representation”. In: *In EMNLP*. 2014.

[17] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier”. In: *CoRR* abs/1602.04938 (2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

[18] Miguel Romero Calvo. *Dissecting BERT Appendix: The Decoder*. Ed. by Medium. URL: https://medium.com/dissecting-bert/dissecting-bert-appendix-the-decoder-3b86f66b0e5f.

[19] Miguel Romero Calvo. *Dissecting BERT Part 1: The Encoder*. Ed. by Medium. URL: https://medium.com/dissecting-bert/dissecting-bert-part-1-d3c3d495cdb3.

[20] Wojciech Samek and Klaus-Robert Müller. "Towards Explainable Artificial Intelligence". In: *Lecture Notes in Computer Science* (2019), pp. 5–22. ISSN: 1611-3349. DOI: 10.1007/978-3-030-28954-6_1. URL: http://dx.doi.org/10.1007/978-3-030-28954-6_1.

[21] Ramprasaath R. Selvaraju et al. "Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization". In: *CoRR* abs/1610.02391 (2016). arXiv: 1610.02391. URL: http://arxiv.org/abs/1610.02391.

[22] Aditya Thiruvengadam. *Transformer Architecture: Attention Is All You Need*. Ed. by Medium. URL: https://medium.com/@adityathiruvengadam/transformer-architecture-attention-is-all-you-need-aeccd9f50d09.

[23] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[24] Francesco Ventura and Tania Cerquitelli. "What's in the box? Explaining the black-box model through an evaluation of its interpretable features". In: *arXiv e-prints*, arXiv:1908.04348 (July 2019), arXiv:1908.04348. arXiv: 1908.04348 [cs.CV].

[25] Jesse Vig. "Visualizing Attention in Transformer-Based Language Representation Models". In: *CoRR* abs/1904.02679 (2019). arXiv: 1904.02679. URL: http://arxiv.org/abs/1904.02679.