POLITECNICO DI TORINO

Master of Science in Computer Engineering

Politecnico
di Torino

Master Degree Thesis

Explaining bias in modern Deep Language Models

Supervisors

Prof. Elena BARALIS

Dr. Giuseppe ATTANASIO

Candidate

Christian Vincenzo TRAINA

April, 2022

Abstract

In recent years, episodes of hate speech on the Internet have increased. Hate speech manifests with instances of misogyny, racism, and attacks on minorities. To analyze large amounts of data and curb the spread of hurtful content, modern language models such as BERT are currently employed in the task of automatic hate speech detection.

Although these models have outperformed previous solutions, several recent works have shown that they still suffer from unintended bias. Biased models tend to be over-sensitive to a limited set of words, so they base the entire decision on only those words and ignore the context.

Much recent work has focused on explaining the models, on the overall understanding of the output, and the way it is obtained. Explanation methods can be based on either exploiting the inner workings of the neural network or analyzing the output by perturbing the input.

In this thesis, several techniques are used to explain neural networks, including attention maps, as extracted from BERT inference; their transformation, called effective attention maps; Hidden Token Attribution, which is a gradient-based explainer; a hierarchical explainer called Sampling-And-Occlusion (SOC); Minimal Contrastive Editing (MICE), which is a modern algorithm that uses counterfactual explanations; and two different SHAP versions: KernelSHAP and DeepSHAP.

The main contributions of this thesis concern the selection of the best explanation methods for the detection of unintended biases in modern neural networks, evidence that most explainers express different types of explanations, and evidence that peaks in contribution scores are more common in false-positive samples.

The analysis was performed on two different hate speech detection datasets, both in English. The datasets were collected from Twitter and manually annotated. In particular, they concern misogyny and hatred against immigrants.

The explanations have been evaluated according to their quality, measured by the deviation from human explanation results. The latter was determined by a survey in which 25 volunteers were asked to indicate the most important words in a sentence. Other parameters included lead time, ease of reading, and theoretical background.

The results on these datasets show that the attention maps and the effective attention maps express the same type of explanation in most of the cases considered. The quality is very high in both of them, even if the output is not easily understandable by non-insiders, since it requires technical knowledge about the network. In MiCE and SOC, thanks to their particular outputs which are respectively hierarchical and contrastive, bias can be quickly individualized, so their use

is encouraged despite their high lead time. Moreover, HTA is fast to compute and its quality remained consistent across experiments. Finally, both DeepSHAP and KernelSHAP were able to detect the bias in most cases, but the quality of the explanation was significantly lower compared to the other methods.

Two additional experiments were conducted to prove whether the presence of peaks on the word contribution score - derived from the attention features - was greater in false-positive samples and whether there was a correlation between the explainers used. The results of these experiments show that there is a correlation between false-positive samples and peaks in attention features and that none of the explanation methods used is redundant, with the sole exception of attention maps and effective attention maps.

Acknowledgements

Prima di procedere con la trattazione di questa tesi ci tengo ad utilizzare questo spazio per ringraziare tutti coloro che mi sono stati vicini in questi anni.

In particolare ringrazio Mikela, Cristina, Andrea e tutti gli amici che mi hanno sopportato e supportato, tutti i colleghi che ho conosciuto e che hanno fatto la differenza, sia aiutandomi materialmente che offrendomi qualche parola e che, nel complesso, hanno reso questo percorso meno ripido. Un ringraziamento speciale a Giuseppe Attanasio che con pazienza e competenza mi ha accompagnato lungo la stesura di questa tesi.

Infine, ringrazio Fabio Fasano e l'azienda per cui lavoro, che da sempre mi ha offerto la flessibilità e il supporto necessario per portare a termine questo obiettivo.

...a mia zia, che non è riuscita a vedermi laureare,
ma che immagino fiera di me.

Table of Contents

List of Tables VI

List of Figures VII

Acronyms X

1 Introduction 1
- 1.1 Thesis objective . . . 1
- 1.2 Organization of the elaborate . . . 5

2 Background 6
- 2.1 Introduction to Natural Language Processing . . . 6
- 2.2 Attention based networks . . . 7

3 BERT and the hate speech context 13
- 3.1 BERT: a Transformer based model . . . 13
- 3.2 Hate speech and biased networks . . . 17

4 Related works 19
- 4.1 Evaluation of intepretability approaches . . . 19
- 4.2 Attention as explanation . . . 21
- 4.3 Recurrent patterns in attention maps . . . 24

5 Overview of explanation techniques 27
- 5.1 Explanation in Machine Learning . . . 27
- 5.2 Explanation taxonomy . . . 29
- 5.3 LIME . . . 30
- 5.4 Shapley values . . . 30
- 5.5 SHAP . . . 30
- 5.6 MiCE . . . 33
- 5.7 SOC . . . 33

6 Methodology 36
6.1 Approach 36

7 Experiments and Results 42
7.1 Dataset 42
7.2 Training 43
7.3 Selection of sentences 44
7.4 Comparison of explainers 46
7.4.1 Human explainer 46
7.4.2 Attention 50
7.4.3 Effective attention 55
7.4.4 SOC 58
7.4.5 HTA 64
7.4.6 KernelSHAP 66
7.4.7 DeepSHAP 70
7.4.8 MiCE 71
7.4.9 Comparison among explainers 74
7.5 Analysis of vertical patterns 82
7.6 Correlation among explainers 85

8 Conclusions and further steps 90

Bibliography 92

List of Tables

7.1 72
7.2 73
7.3 73
7.4 MiCE explanation 80
7.5 Lead time to obtain execution. Time expressed in seconds 80

List of Figures

2.1 Example of attention map from the paper "NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE" that firstly introduced attention. In the picture is visible where the neural networks focus to translate a sentence from english to french 9
2.2 The transformer architecture, taken from "The illustrated transformer" 10
2.3 The transformer encoder, taken from "The illustrated transformer" . 11

3.1 Example of a masked token. "Do" was marked as changed but it was not, "The" is marked as changed and it actually was, and another random word was masked out 14
3.2 Architecture and example of working of BERT. Taken from the paper "On the Role of Conceptualization in Commonsense Knowledge Graph Construction" 16

4.1 Example of the 5 recurrent patterns, taken from the original paper . 26

5.1 Example of hierarchical explanation 34

7.1 Example question of the survey 47
7.2 Result for the question referring the sentence "Fuck yeah you pussy boy" 49
7.3 Result for the question referring the sentence "@benshapiro Fuck you pussy" 49
7.4 Answers to the question "@minimaslany What if she enjoys it? I think the days where women are forced to stay in the kitchen, not work, etc are over." 50
7.5 Aggregated importance given to the words of the sentence "@minimaslany What if she enjoys it? I think the days where women are forced to stay in the kitchen, not work, etc are over." 51
7.6 52
7.7 53

7.8 54
7.9 55
7.10 Raw attention 57
7.11 Effective attention 58
7.12 59
7.13 60
7.14 a) Pearson correlation index between effective attention and raw attention, over different token lengths. b) and c) respectively raw and effective attention, where each point represents the average attention of a given head to a token type. Figure taken from [30] . . 60
7.15 62
7.16 62
7.17 63
7.18 *Note: the sentence was split into two different images for readability reasons* 63
7.19 65
7.20 65
7.21 66
7.22 66
7.23 68
7.24 68
7.25 69
7.26 70
7.27 71
7.28 Attention map of the 11th layer, averaging across heads 75
7.29 Effective attention map of the 11th layer, averaging across heads . . 76
7.30 SOC explanation 76
7.31 HTA explanation 77
7.32 DeepSHAP explanation 79
7.33 KernelSHAP explanation 79
7.34 The same attention map before and after increasing the contrast . . 83
7.35 Plot of the normalized vertical pattern counts varying the parameter β 84
7.36 Plot of the normalized vertical pattern counts, focusing on false positives and false negatives, varying the parameter β 85
7.37 Plot of the difference of normalized vertical pattern counts related to false positive and false negative, varying the parameter β 86
7.38 88

Acronyms

XAI
Explanation AI

NLP
Natural Language Processing

BERT
Bidirectional Encoder Representations from Transformers

NN
Neural Network

AI
artificial intelligence

RNN
Recurrent Neural Network

BiRNN
Bidirectional RNN

LSTM
Long Short Term Memory

MLM
Masked Language Model

NSP
Next Sentence Prediction

CLS

Classification token

SEP

Separator token

GDPR

General Data Protection Regulation

LOO

Leave One Out

Chapter 1

Introduction

1.1 Thesis objective

Online platforms are a fertile field for hate speech and attacks on minorities. According to a 2018 report from the University of Nottingham, 21.68% of women surveyed reported being sexually assaulted online, while 54.30% received unwanted sexually explicit texts. This resulted in, among other things, 1.7% of respondents speaking less online.

Companies offering a similar service know the problem and try to moderate their platform to remove toxicity, but since the explosion of social media in recent years, this task can no longer be done manually. Because of this, companies have been using artificial intelligence to automatically address the problem by either automatically making a decision or flagging content as needing moderation. Many different architectures have been proposed to achieve this goal. The oldest and most shallow solutions involved the use of decision trees and rule-based classifiers, which were able to detect the most offensive words, but were unable to generalize sufficiently and perform well in unseen scenarios, and were also easily fooled.

A major performance improvement in toxicity detection systems was achieved through Deep Learning. With Deep Learning, classifiers were able to tailor their classification to context simply by using large datasets of examples extracted from the platform. The first example of a neural network designed specifically for sequence classification was the Recurrent Neural Network (RNN). They were soon replaced by Long-Short Term Memory (LSTM), which solved many of the problems of RNNs. Nowadays, most toxicity detection systems are instead based on BERT or one of its variants. BERT is an architecture introduced by Google that does not use recurrence to examine and classify the sentence. Instead, it uses an attention-based mechanism that allows tokens to be related to each other at any distance in a sequence and assign them a weight.

Attention was proposed by Google in 2016 when it introduced a new architecture called the Encoder-Decoder [1], which makes extensive use of attention mechanisms to improve performance. Attention was proposed to solve a particular problem that occurred when encoding the input sequence into a fixed-length vector and then decoding the vector to obtain the desired output. It is believed that this problem is more likely to occur when decoding long sequences. The attention mechanism was then adopted by the transformer architecture, which made still heavier use of attention so that the original paper was named "Attention is All You Need". In the paper, the authors use attention to relate words to each other, defining a square attention matrix that they call self-attention. Self-attention relates different tokens of the same sequence to each other to compute an overall representation. In 2018, Google introduced BERT, an evolution of Transformers that eliminates the decoder and uses only 12 stacked encoders. BERT has dramatically increased accuracy and is considered state of the art for many NLP tasks. Since the decoder is omitted, a new token is prepended in each sequence to allow classification. This token is called CLS and is the main output of the neural network. Another output of the neural network is the attention matrix, which is used for computation. In fact, each token is related to every other token to get the meaning of the sentence. A token that is attended by all the other tokens could be considered more important than a token that is not attended by any of the other tokens since the former has been weighted more than the latter in the computation. For this reason, we can intuitively view the attention maps as a metric for the importance of tokens in a sequence.

In 2019, the paper "Attention is not explanation" was published, stating that attention is not a valid explanation metric and is not sufficient to explain an attention-based neural network. In the same year, another paper titled "Attention is not not explanation" was published, contradicting the previous work. In this paper, however, attention is not used as an explanation metric but is explored in the context of bias detection. In August 2019, the paper "Revealing the Dark Secrets of BERT" was published, analysing the patterns that attention maps adopt. One of the findings of this research was that the patterns of attention maps can be described in 4 categories: heterogeneous, block, vertical, and diagonal. Therefore, an investigation was conducted to see if any of these patterns were associated with bias within a fine-tuned BERT model.

Indeed, for this purpose, a dataset of 3600 sentences was used to check the spot the patterns and eventually relate with the hate speech context. The sentences were manually labelled as misogynistic or not and were extracted from the Twitter platform over a long period of time. In this thesis, attention was not only analysed in its raw version, as it was extracted from the neural network, but also after two different transformations, namely effective attention and entropy attention. The first is a measure of the attention that effectively affects the decision, the second is a measure of how much the attention weights are distributed around a limited set

of tokens.

Altogether to the analysis performed over attention maps, different classical explanation methods were used to interpret and detect bias within the neural network. While the attention-based explanatory algorithms are model-specific and task-specific, other algorithms used for the comparison, instead, are classified as model-agnostic since they only act on the input and on the output to produce an explanation. Among the galaxy of algorithms available, they were selected **MiCE**, which provides an explanation by producing edits that are contrastive, minimal, and fluent. In this way, it allows the user to focus on the words that were determining the decision, and **SOC** which determines importance as the difference between the expected prediction after masking part of the sentence, and the expected prediction before the masking, for each replacement of contexts. For the aforementioned algorithms, an implementation for BERT was already available and was used for the scope. Instead, other model-agnostic algorithms have needed a custom implementation, such as **HTA**, which stands for Hidden Token Attribution and it is a variant of the gradient attribution algorithm for BERT tokens, or SHAP, which uses a coalitional game theory approach to understand how a coalition of features affects the output.

For the purpose of the experiments, BERT was fine-tuned using 4 epochs over the train set, it reached an accuracy of 72.3% over the test set. Then 15 sentences were selected from the two used datasets. Particularly, 10 sentences were extracted from the misogyny test set, 7 false positive samples, 3 false negative samples, and 5 sentences have been extracted from the hate speech against immigrant datasets, respectively 3 false positive samples and 2 false negatives samples. The sentences were selected to see how they relate with the explanation methods, and they have been selected using some parameters that aim to cover all the possible shades of false positive and false negatives, such as rhetorical questions, sarcasm, neutral words in a toxic context or stand-out toxic words used in a supportive context.

Then an evaluation of the different explainers on the 15 selected sentences was proposed, in order to obtain insights on the way the classification is performed and why some mistake was committed by the neural network, such as false positive or false negative. Also, the parameters used for the evaluation of the explanation algorithms were defined, and they include the quality of the output, the ease in its reading, the lead time, and the theoretical background.

To remove as much subjectivity as possible from the research, a survey has been proposed to 25 volunteers, and they were asked to provide an explanation to the selected sentences in the form of the selection of a group of words that led the decision. The selected words, for each sentence, were aggregated and used to assign an importance score. Thus, the results from the human explanation survey have been used as a benchmark for evaluating to which degree the neural network was committing a mistake, and what was the optimal representation that it had to

reach. Downstream of the analysis performed, all the explainers were able to detect the bias, with different levels of quality.

Attention maps and effective attention maps were revealed to be very similar and provide the same explanation, but the quality has been found to be very high, even if not easily read from not insiders. Thanks to its fast output, it can be effectively used for the early detection of biases during the development of the neural network. MiCE and SOC used two innovative approaches for the production of the explanation, the first generated contrastive sentences while the second represented the input in a hierarchical structure to appreciate the impact of combinations of tokens. Both of them reach high quality in the detection of the bias.

Furthermore, MiCE has been demonstrated to be able to detect a series of biases in the trained neural networks also in the cases that were not strictly related to the task that was being analysed. Indeed, since its generative nature, it was able to generate sentences that contained elements of hate speech against minorities also in the model trained to detect misogyny.

HTA is another fast algorithm that attributed an importance score to each input token accordingly to the gradient information used for the inference. This algorithm demonstrated to be able to provide insights that other explanation methods were not able to show.

Finally, both DeepSHAP and KernelSHAP were able to detect the bias in most of the cases, but the quality of the explanation resulted to be sensitively lower when compared with other methods. Two further experiments have been executed for this thesis, one aimed to prove whether the presence of vertical patterns in effective attention maps was greater in false positive samples or not. Thus, an algorithm that counts the number of vertical patterns for, respectively, false positive, false negative, true positive and true negative samples has been developed, the count has been then normalized and plotted on a graph varying some parameters of the algorithm. The result of the experiment showed how there are, on average, 20% more vertical patterns in false positive samples than in false negative samples. These results motivate the use of all the regularization techniques that aim to smooth peaks in attention maps, since in the experiment has been reported that the false positive rate and vertical patterns rate have a slight positive correlation.

The last experiment aimed to demonstrate whether there is a correlation between the used explainers, with the objective of detecting redundant explainers that provide the same explanation.The output of all the used explanation algorithms has been manually annotated and reported in a file, thus a correlation index has been calculated for each couple of algorithms, forming a correlation matrix. The result showed that there is not a correlation among the used explainers, with the only exception of attention maps and effective attention maps.

Disclaimer: This thesis, because of the datasets related to the context of hate speech, contains words or concepts that many might find offensive or hurtful.

However, their use in explicit form cannot be avoided without altering the meaning of the work. For this reason, it was decided not to censor any of the words used.

1.2 Organization of the elaborate

This paper is structured in 6 different chapters, in addition to the current one. In particular, they are organized as follows:

- **Chapter 2:** Background. This chapter introduces the concepts of NLP and the recent deep learning approaches, briefly retracing the history. The chapter offers an introduction to dive into the complex topic and covers all the most important theoretical aspects.
- **Chapter 3:** BERT and the hate speech context. This chapter offers a broader discussion of attention-based architectures and how they work internally. Following, it discusses what is meant by bias in the context of hate speech.
- **Chapter 4:** related works. In this chapter will be inspected the related works deemed important for the thesis. Particularly will be analysed the papers "Attention is not explanation", "Attention is not not explanation" and "Revealing BERT's dark secrets".
- **Chapter 5:** overview of explanation techniques. There are different types of explanation techniques and many more are invented every year. In this chapter, what's meant by "explanation" is defined more precisely, and all explanation techniques are presented and explained.
- **Chapter 6:** Methodology. in this chapter the methodology used for the research is explained. The structure of the experiments is broken down, and the reasons that led to the choices for the analysis are explicated.
- **Chapter 7:** Experiments and results. In this chapter the datasets used and the way the neural network has been trained is illustrated. Thus are reported the experiments performed and explained how to reproduce them, altogether with the results and their discussion.
- **Chapter 8:** Conclusions and further steps. In this chapter, the results obtained from the experiments are summarised, and some further steps for the improvements are proposed.

Chapter 2

Background

In this chapter a brief understanding of the Natural Language Processing task and its evolution is offered, including the outbreak of attention that brought to even more performant and complex architectures.

2.1 Introduction to Natural Language Processing

By natural language processing (NLP) we mean all applications of artificial intelligence that bridge the communication gap between computers and humans. The need for an automated natural language processing tool is as old as the Cold War, when the U.S. developed a Russian-to-English translation system in 1960, defining one of the first examples of machine translation. NLP is divided into two main branches: natural language generation, which involves generating text and predicting the next word or phrase in a corpus, and natural language understanding, which, as the name implies, involves understanding a language and is applied, for example, to classification tasks. In the past, natural language processing required the help of various experts to perform well, since phonology, morphology, and syntax were all features of a language that had to be interpreted and inserted manually. Thanks to the great advances in artificial intelligence in recent years, modern neural networks are able to extract this information automatically, making the process cheaper and easier, but in contrast, requiring a huge amount of data. One of the first examples of Deep Learning applied to NLP was RNN. RNN stands for Recurrent Neural Network and is a type of neural network that is particularly suited for sequential data. The architecture of an RNN is similar to that of a feed-forward neural network, but the RNN simulates a memory that is able to remember previous states by using recursion. It works by passing information from the previous hidden state to the current state, and in total to the new input example of the sequence. In this way, the neural network uses for its prediction

not only the information of the current input but also all the information coming from the beginning of the state.

Even though they were state of the art when they were introduced in 1986, they still have some problems, namely the exploding gradient and the vanishing gradient. While the exploding gradient can be mitigated by truncating the gradient above a certain threshold, the vanishing gradient posed a more serious problem and led to the development of new concepts, such as the LSTM, in 1997. The vanishing gradient problem was caused by the iterative application of the softmax function, which caused the gradient to become thinner and thinner during backpropagation, so that for long sequences it happened that the last element could no longer affect the neurons in the older part. The LSTM solved this problem by introducing gates that can open and close depending on certain values that are part of the learning process, and that are able to capture the signal and keep it alive so that it does not vanish. In other words, the gates decide which data is important and can be useful in the future and which data must be deleted. The three gates are the input gate, output gate, and forget gate.

The LSTM solved many of the problems that RNN had, but an additional performance boost occurred when attention mechanisms were applied to the LSTM. Attention is the idea of freeing the encoder-decoder architecture from the fixed-length internal representation. This is accomplished by keeping the intermediate outputs of the encoder-LSTM from each step of the input sequence and training the model to learn to pay selective attention to these inputs and associate them with elements in the output sequence. In other words, each element in the output sequence is dependent on certain elements in the input sequence.

For a deeper introduction to attention and attention-based architecture, see the next paragraph.

2.2 Attention based networks

Attention was firstly introduced by Google in 2016, in the context of sequence-to-sequence algorithms. Deep learning models as sequence-to-sequence are capable of carrying out a wide range of tasks, such as machine translation, text summarization, and image captioning. The typical examples of sequence-to-sequence models are those where the task is to translate from one language to another. For this reason, at the end of 2016, Google updated its Google Translate service with a Neural Translation Machine, which caused a major impact on performance.

These models are explained in the two pioneering papers, the first one was named "Sequence to Sequence Learning with Neural Networks" where Sutskever et al. introduce the first sequence-to-sequence algorithm using LSTM and distinguishing an encoder part and a decoder part, and making minimal assumptions on the

sequence structure. Firstly, an LSTM is used to map the input sequence to a vector of fixed dimensionality, and then another deep LSTM is used to decode the target sequence from the vector. The second paper was named "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation" where Cho et al. introduces the encoder-decoder architecture, an architecture that is composed of two recurrent neural networks stacked one upon the other. One RNN encodes a sentence into a fixed-length vector while the other one decodes the vector into another sentence. The encoder and decoder of this model are thus trained together to maximize the probability of a target sequence given a source sequence.

The task of neural translation can be formally defined as the model that maximise the probability that, given a certain sentence in a language, we find the corresponding sentence in the other languages. Using a mathematical approach, the same task can be defined using the following formula:

$$argmax_y P(y|x)$$

Where $x \in X$ is a random sentence extracted from the source language, while $y \in Y$ is a sentence in the target language. Using the Bayes rule, the formula can be rewritten as:

$$argmax_y P(x|y)P(y)$$

In this way, the problem has been split into two sub-problems. It is important to notice that the values that maximize the probability of finding a x given y are known, and the model and its values can be called "translation model". The latter represents the probability that y is effectively a sentence, which we can name the language model.

The Neural Translation problem can be solved using a sequence-to-sequence model, which is a model that takes a sequence of items and outputs another sequence of items. In this case, any item is a word belonging to a dictionary, either from the source language or the target language. As stated before, the model is composed of an encoder and a decoder: the encoder processes the input sequence and compiles it into a vector, then the output item is produced by the decoder. The context vector can be set to a size that is consistent with the number of hidden units in the RNN, and some of the values that it can assume are 256, 512, or 1024. An RNN takes two inputs at a time: one for the item that is being processed in that given time step, and one for the hidden state, which is obtained from the previous time step. Indeed, in the previous time step, the decoder had maintained a hidden state that it passes from one-time step to the next. Since neural networks only deal with numerical values, a word embedding algorithm is needed to transform words into vectors. The encoder processes the input sequence and compiles it into a

vector, which is also named the context. This context may have any size, thus the size is a hyper-parameter of the model. After that, the encoder sends the context over to the decoder, which begins producing the output sequence item by item. The encoder and decoder may be both be either RNN or LSTM.

According to many pieces of research, among all Bahdanau et al. [2] and Luong et al. [3], the context vector has been proved to be the bottleneck for these types of models, thus when dealing with long sentences it was challenging to maintain a reference (in other words "remember") what was the subject in the beginning. To patch these problems, the mentioned papers introduced or refined a technique called "Attention", which had a big impact on improving the quality of machine translation systems. Thanks to its internal working, attention allows the model to focus on the relevant parts of the input sequence as needed since the most important words receive higher importance and do not get easily forgotten.

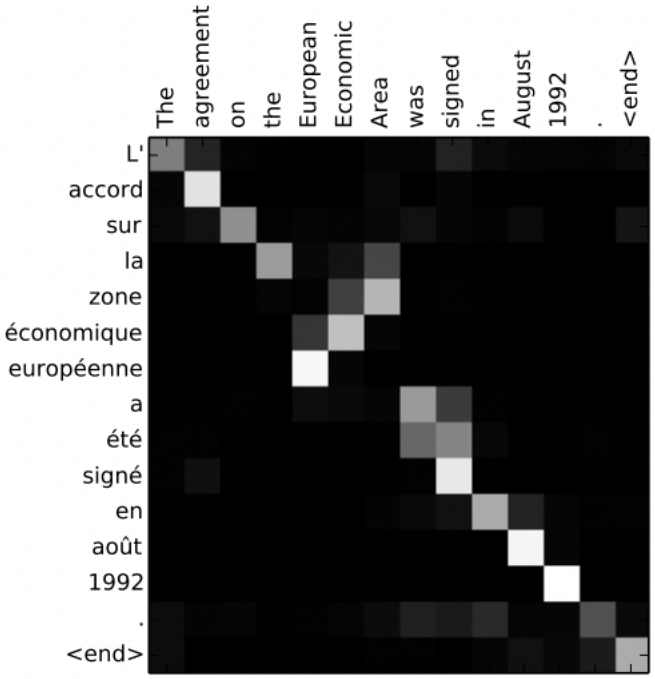

Figure 2.1: Example of attention map from the paper "NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE" that firstly introduced attention. In the picture is visible where the neural networks focus to translate a sentence from english to french

The encoder-decoder architecture with attention is very similar to the original one. The change in the architecture only concerns the decoder which is extended

with an attention mechanism. The attention mechanism is implemented as weight attributed to each part of the input. The weights are learnt through the gradient descent algorithm, and a soft-max transformation is applied to provide some properties as additivity and make them sum to one. This allows the decoder to only focus on the words that are really relevant to the output.

The attention mechanism improved the performance and made the encoder-decoder architecture the new state-of-the-art in these type of tasks. Furthermore, it inspired new architectures that were entirely based on attention. An example of this, is the paper published by Vaswani et al. and named "Attention is all you need" [4]. In the paper, the authors introduce a new architecture that entirely relies on attention, which got the name of Transformer.

The Transformer is a model that removes the concept of recurrence and only relies on attention, leading to a boost in the speed in almost all the cases where it was applied to existing neural networks. In many cases, the performance obtained from attention-based networks outperformed the Google Neural Machine Translation model, which was considered state-of-the-art before the introduction of the Transformer architecture. The reason why such a good performance has been reached lies in the ease with which the Transformer lends itself to parallelization.

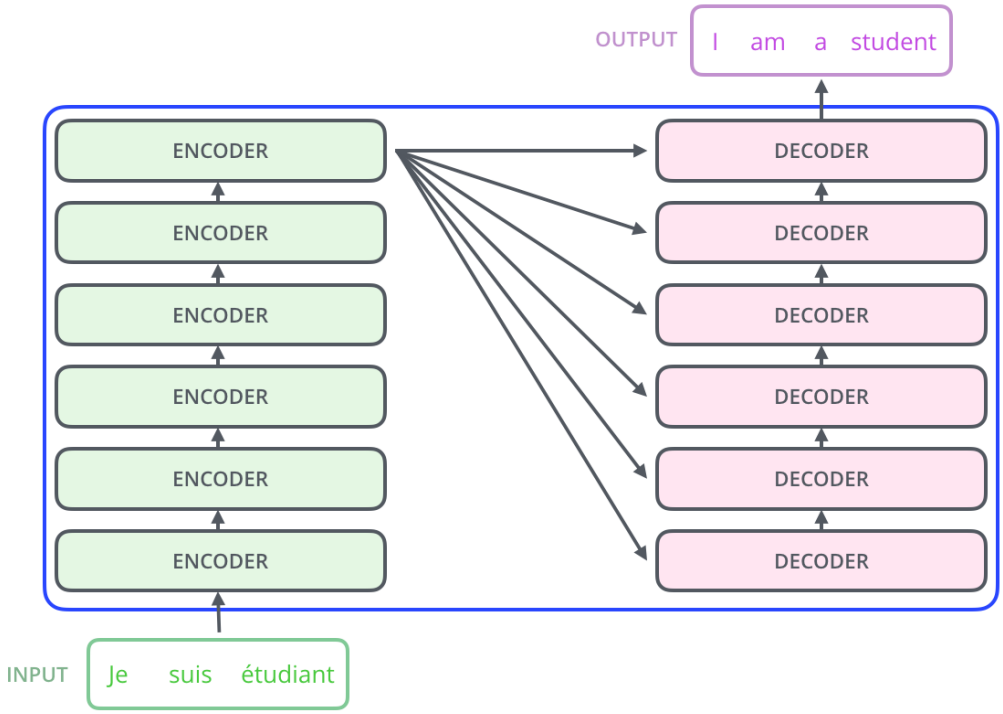

Figure 2.2: The transformer architecture, taken from "The illustrated transformer"

As it is visible in the Transformer architecture figure visible in the figure 2.2,

the whole architecture can be broken down into a stack of encoders and a stack of decoders. The encoder sees the presence of a new layer, named self-attention layer, which goal is about helping the encoder look at other words in the input sentence and encoding to a specific word. The output is then fed to a feed-forward neural network, which is trained altogether with other weights and performs a transformation over the output vector, aligning it to the next encoder or to the classification layer in case it is the last feed-forward layer. All the layers of the encoder have the same role, that is accepting the output of the previous encoder and performing the same computation. The only exception is the first encoder which works on a word embedding, and not on the previous layer output.

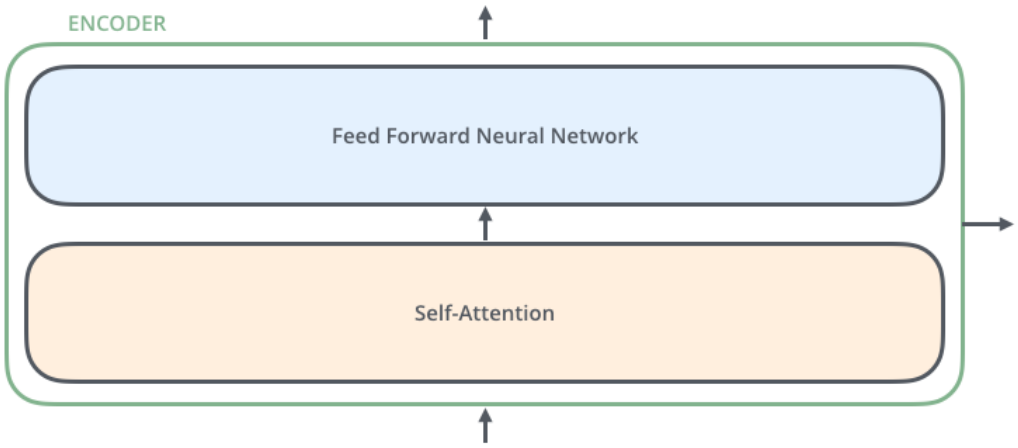

Figure 2.3: The transformer encoder, taken from "The illustrated transformer"

Self attention is calculated through the creation of three vectors for each word embedding, these vectors are namely the Query vector, the Key vector and the Value vector. These vectors are created by multiplying the embedding by three matrices that were learned by back-propagation.

These vectors are used to calculate a score, which is calculated by taking the dot product of the query vector V with the key vector K of the respective tokens. Therefore, in the processing the self-attention for the word in the i-th position, the first score is calculated as the dot product between q_i and k_i. Thus, the second score is calculated as the dot product of q_i and k_{i+1}, and so on. The original paper, thus, advises dividing the score by 8, which is the square root of the dimension of the key vector, which instead was 64. This division was performed to have a more stable gradient, as explained in the appendix of the paper. After the division, the

score is transformed using the soft-max function, in order to get a normalized set of values, and make the scores sum to 1. Another desired property, derived from the use of the soft-max function, is to have a vector that looks like a probability distribution.

Finally, each value contained in the value vector is multiplied by the obtained score. In this way, the result is similar as each value is assigned a weight, which corresponds to the score calculated previously. Furthermore, as a consequence of the use of the soft-max function, each of the calculated scores is in the range between 0 and 1.

Thus, to obtain the desired attention map, the weighted value vectors are summed up. In the equation 2.2 all the steps mentioned are formalized.

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

The original paper also refines the self-attention mechanism, by introducing a further concept which is called "multi-headed attention". The multi-headed attention mechanism works similarly as what has been described above, but the same process is repeated for a number of times that is proportional to number of heads. This refinement leads to some advantages, firstly it expands the model's ability to focus on different positions. Although each layer contains a bit of every other encoding also in the previous definition, it could be dominated by the actual word itself. Secondly, it gives the attention layer multiple representation subspaces. Indeed, repeating the process a number of times, thus using several heads, the neural network will contain multiple sets of Query/Key/Value weight matrices. Since the Transformer architecture uses eight attention heads, it will end up having eight sets for each encoder/decoder. On the other hand, the BERT architecture uses 12 heads for each layer, thus 144 heads are contained across the whole architecture. The only variation between one head and another consists in its random initialization. Then, after training, each set is used to project the input embeddings into a different representation subspace.

Chapter 3

BERT and the hate speech context

3.1 BERT: a Transformer based model

BERT came out from the Google research team in October 2018 [5], and it immediately achieved impressive results on several difficult natural language tasks. Indeed, other independent researchers claimed that the performance of a variant of BERT surpassed human-level performance on some specific tasks [6].

BERT stands for **Bidirectional Encoder Representations from Transformers** and, as the name says, it is a bidirectional model that makes large use of Transformer encoders. Thus, the architecture of BERT is very similar to the one already seen in the encoder-decoder model, the main difference is that the decoder part has been dropped in this new setting, and the number of layers has been increased from 6 to 12. Other main differences that disrupted the NLP community are not related to the architecture, but to the ways it has been trained and to the ways it can be utilized for custom tasks. BERT utilizes the concept of transfer learning in order to let users customize the neural network for the specific case, avoiding to re-train the weights. Indeed, the model is very large and this can be seen as a drawback for many applications since a lot of data and a lot of expensive hardware are needed and this could be prohibitive for many little to medium realities. On the other hand, the BERT pre-trained model has been made available by Google, and it is thus possible to take the model and apply little changes to it, by changing the output layers and performing a fast fine-tuning training, in order to have state-of-the-art performance in custom specific tasks. For this reason, we can distinguish two different phases to create a trained custom model, and they are **pre-training** and **fine-tuning**.

During the pre-training phase, the model is trained on unlabeled data over

different pre-training tasks. On the other hand, in the fine-tuning phase, the BERT model gets firstly initialized with the pre-trained weights, and all of the parameters are fine-tuned using labeled data from the downstream tasks. A distinctive feature of BERT is its unified architecture across different tasks. There is minimal difference between the pre-trained architecture and the final downstream architecture.

The BERT, as the first word of the acronym says, makes from the fact of being bidirectional one of its strengths, indeed the architecture does not only look for the next word in a sentence to make a sense, but it also looks for the previous word. BERT was not the first bidirectional architecture for NLP, ELMo (Embeddings from Language Models) [7] was previous in time and has earned credit for its language understanding skill and for the fact it was able to understand the context in a sentence. However, ELMo reached bidirectionality by focusing on the forward words and backward words in two different time steps, on the other hand, BERT is able to look at all the context it needs whenever it needs, since it has access to all the words in every time step and it can use attention to focus only on the relevant ones.

To train the architecture in a way it can understand which words are valid for the context, BERT is trained on a specific language in a totally unsupervised manner. Since the large number of weights, a lot of train samples are needed to reach a good fit of the network. However, we just need to provide a large corpus in a specific language, like a set of books or the Wikipedia dump, and this will be enough to learn the basic rules of a language. The Google researchers have identified two different tasks to reach a good understanding of the language in a bidirectional way.

The first task is named **Masked Language Model** and it is achieved by masking part of the input. BERT was fed with the data coming from Wikipedia and the BookCorpus, but 12% of the words were randomly chosen to be masked out with a special token without any particular meaning, and BERT was trained to reconstruct the original sentence by finding the missing words. In addition to masking 12% of the words, the pretraining was performed by replacing 1.5% of the words with another random word randomly selected from the corpus, and another 1.5% was marked as changed but was not actually changed. The total amount of the abnormal words during the pretraining sums up to 15%.

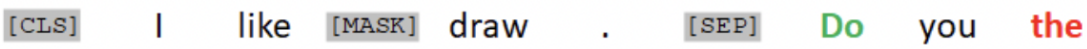

Figure 3.1: Example of a masked token. "Do" was marked as changed but it was not, "The" is marked as changed and it actually was, and another random word was masked out

The reason why the Google researchers did not limit themselves to just masking

out the words is that the mask token does not provide any meaning of the word it replaced, BERT has to infer what word was there purely by looking at the context. This might give BERT a tendency to ignore the input embedding and infer everything from its context, which is often not the desired outcome. According to the appendix of the original paper, the percentages 12%, 1.5% and 1.5% have been obtained through repeated experiments.

The second task takes the name of **Next Sequence Prediction**. In this task, BERT was fed with pairs of text passages. In 50% of times, the second passage immediately followed the first one, while in the other 50% the second passage was just randomly taken from elsewhere in the training text. According to the appendix of the original paper, MLM and NSP are actually executed concurrently. One training sample consists of a chunk of text, with all of these various substitutions made, and this results in a number of different predictions that need to be made. The training loss is the sum of the mean masked LM likelihood and the mean next sentence prediction likelihood.

In this discussion, the locution "word" was used to indicate the piece of information that BERT works on. Actually, BERT utilizes a subpart of a word, which takes the name of token. BERT uses a limited dictionary of words or subwords, each item in the dictionary corresponds to a token. If the word is common in the used language, chances are it uniquely corresponds to a single token. On the other hand, if the word is not common, it is a composed word or it is a declination of an original word, chances are it gets split into a set of tokens. As an example, the word "assistant" can be divided into the tokens "assist" and "ant". A token comes out from the built-in BERT Tokenizer, which will take the raw text string and break it into tokens that BERT can be feed with. The reason BERT provides its own tokenizer is that it has its own fixed vocabulary of tokens, with an embedding associated with each of these. Indeed, BERT model has a fixed vocabulary size of about 30,000 tokens total. Somewhere between 60-80% of these are whole words, and the rest are subwords.

If a word is not in BERT's vocabulary, then it simply breaks it down into subwords. The subword can be as little as a single character, in an extreme case. The reason why BERT utilizes a so small vocabulary, in contrast with other previous solutions, is that rarer form of the same semantic word can be induced from the more common forms, and in this way BERT can make a sense of words even if it has never seen them before.

As discussed before, BERT reaches bidirectionality by looking all the tokens at the same time. However, in this way, there is the risk that a piece of information is missing, that is the position of a token within the sentence. The position of a token within the sentence is important to understand the overall meaning, for this reason a progressive cyclical number, obtained from combination of trigonometric functions, is included while creating the embedding of the token.

Among all the tokens obtained from words or subparts of words, there are three special tokens used by BERT for its inner working. They are *[CLS]*, *[PAD]*, and *[SEP]*. The [CLS] token is useful to indicate BERT we are performing a classification task. The CLS token will contain the output of our classification, once going through all the 12 levels of BERT. Instead, the [PAD] token is used for padding while the [SEP] token is used to separate two sentences.

To understand how the token [CLS] allows the classification, a step back is needed. During the NSP task, Google researchers added a simple linear classifier in the last layer of BERT, and only the final embedding for the [CLS] token was fed into it. Using gradient descend to optimize the problem, BERT has learned that to perform well on this task, it needed to enrich the [CLS] embedding with all of the information it would need to make the classification decision. For this reason, BERT always expects the [CLS] token as the first token, and it will then enrich the [CLS] embedding with information about the whole input text.

As a final note, BERT only works with fixed length sentences. For this reason, if a sentence contains less tokens, all the remaining spaces up to the maximum need to be filled with the [PAD] token.

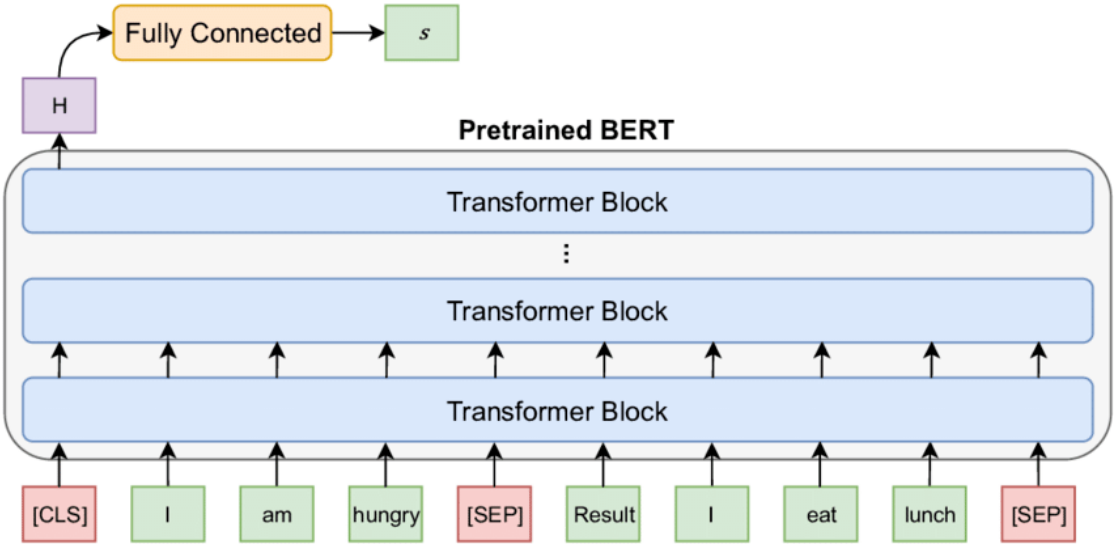

Figure 3.2: Architecture and example of working of BERT. Taken from the paper "On the Role of Conceptualization in Commonsense Knowledge Graph Construction"

3.2 Hate speech and biased networks

The bias in machine learning is a phenomenon that occurs when an algorithm [8], usually a neural network, produces outputs that contain a deviation from the original meaning, thus they are systemically prejudiced due to erroneous assumptions in the machine learning process. In almost all cases, this prejudice is induced by an imbalance in the dataset or by a too shallow classifier. Indeed, the quality of the training depends on the size and on the quality of the training dataset used. Faulty, imbalanced, or approximate datasets will result in inaccurate predictions, and the quality of the output cannot be much better if the quality of the input is kept low. In other words, if a bias is contained in the dataset used for the training, this bias will be transferred to the neural network. In the context of hate speech detection, bias is important because it could afflict categories that are already discriminated. Also, following recent discussions in the mainstream media regarding allegations of racism and sexism assumed by neural networks, researchers have identified the bias in machine learning models as one of the main concerns in the field [9] [10].

The bias into machine learning models is not usually intentional, and it is not introduced by the individuals developing the neural network. The main problem is that the training is executed on human-generated data, and human biases can easily result in a skewed distribution in the training data. Developers and other employees involved in the machine learning field must be proactive in recognizing the biases, to prevent the model to perpetuate the unfairness. However, also what is meant by hate speech, firstly by humans and then for neural network, requires to be defined. According to Warner et al. [11], there are numerous problems in defining hate speech that must be resolved to annotate a corpus and develop a consistent language model.

It is important to consider that the mention of an organization associated with hate crimes cannot in any way be considered hate speech. As in the example, "Ku Klux Klan" in and of itself is not hate speech, as it may appear in historical articles or other legitimate communications. Always according to Warner at al., also an endorsement of the organization does not constitute a verbal attack on another group, thus cannot be considered hate speech. While it is reasonable to assume that such endorsements are made by authors who would also be comfortable with hateful language, in and of themselves these phrases are not classified as hate speech. For the same reason, even excessive pride in his or her own race is not hate speech, since it is not providing an attack in that specific context. Although such boasting may seem offensive and is likely to be accompanied by hateful language, disparagement of others is required to meet the definition. Some research [12] has shown how the use of some words that are frequently used in toxic sentences leads the models over-generalize and disproportionately associate those terms with the toxicity label.

In this way, also a sentence as "I am a gay man" receives unreasonably high toxicity scores.

In a research led by [13], the influence of data bias on the detection of abusive speech is investigated. It is shown that the results reported in previous work for popular datasets are much lower under realistic conditions where these biases are reduced. Such biases are most apparent in datasets created by focused sampling rather than random sampling. Datasets with a higher proportion of implicit abuse are more affected than datasets with a lower proportion. Thus, the way a dataset is composed and how the sentences are selected has a large impact on the biases that will arise in the trained neural network. Hate speech detection gathered a great amount of attention, in order to tame this drift, and new models trained for hate speech detection came out. While many models have claimed to achieve state-of-the-art performance on some datasets, they fail to generalize [14]. As an example, the models may classify comments that refer to certain commonly-attacked identities as toxic without the comment having any intention of being toxic. In other words, the prediction can be based only on a single word, or a handful of them, missing the overall meaning of the sentence. A large prior on certain trigger vocabulary leads to biased predictions that may discriminate against particular groups who are already the target of such abuse. Another issue with the current methods is the lack of explanation about the decisions made. With hate speech detection models becoming increasingly complex, it is getting difficult to explain their decisions. In addition, in 2016 a new European law has been received under the name of GDPR, which among other facts it also introduced the “right to explanation”. This calls for a shift in perspective from performance-based models to interpretable models. As an example, the word "gay" may be usually used in an offensive context, to discriminate against a minority, or as an offensive term. However, if we take the sentence "I am a proud gay", that could be labeled as offensive even though it has not any shade of hate speech. The word is used every day in online language by the LGBT community. Similarly, words like "hoe" and "bitch" are instead used in rap/trap song lyrics. Such language is present on certain social media and any hate speech detection system should include these for the system to be usable. [15]

Chapter 4

Related works

4.1 Evaluation of intepretability approaches

Over the years, various methods have been developed to explain the output of text classification models. With the vast array of explainers, both built-in and post-hoc, it is often difficult to understand what are the main contributions of each of them, in which task it is better to use one than the other, and for which models a particular explainer is best suited.

In their paper, Vivian Lai et al. [16], introduce a methodology to understand how similar the top features are by making a comparison among the models and a comparison among the explainers to provide a systematic characterization of the top features obtained in a way that is as independent of the different factors as possible. The main contributions of their work also include how important features are distributed among instances and into which linguistic cases the found important features fall.

For this purpose, the authors focus on text classification and use a testbed with three tasks. In addition, 4 models are used: Linear SVM, Gradient Boosting Tree, LSTM with attention, and BERT. The first task aims to perform a comparison between the LIME and SHAP explainers. In this case, after running the explainers on both models, Jaccard similarity is applied to the top-k features with the highest absolute feature importance. The use of the top-k features arises from the fact that many models and explainers produce sparse feature importance vectors with many zeros, which could be difficult to handle.

The use of Jaccard similarity has some implications. Different explainers may identify the same set of important features, so Jaccard similarity is high and in this case, the explainers overlap and express the same explanation. However, when Jaccard similarity is low, it is difficult to decide which explainer is used to represent important features. Therefore, the authors exploited these cases to understand

how similarity varies between explainers and models, instances, and features. In other words, the goal is to systematically characterize the similarities between vectors of feature importance generated by different explainers and executed on different models. To make these comparisons, the authors fixed the explainer and compared the similarity value of different models. Conversely, they fixed the model and compared the similarity value of different explainers. The comparison between the models showed that LSTM with attention and BERT obtained two slightly different top 10 features. Regardless of the explainer used, the importance features obtained by SVM and XGBoost were the most similar to each other, and the same pattern has been observed for all the models based on Deep Learning. LSTM with attention and other models that fall under the definition of Deep Learning is the least similar to traditional models such as XGBoost. The same result is obtained for BERT. Another result of this experiment is that post hoc explainers tend to produce similar results that differ from the built-in ones. This is especially true for the similarity between LIME and SHAP and can be explained by the authors by the fact that they are both based on an additive principle.

When the authors fixed the models and tried different explainers, they found that the similarity between important features obtained from different explainers tended to be lower for LSTM with attention. The similarity between the importance of features generated by different explainers is low when LIME is compared with SHAP. However, LIME is not more similar to SHAP than how much it is to built-in explainers, which contradicts an earlier hypothesis of the authors. In summary, post-hoc methods generate more similar important features when comparing different models, but this is not the case when the model is fixed.

In another experiment, the authors investigated how the similarity between instances changes. The result of this experiment is that the similarity between feature importance vectors is not particularly high when two models agree on the predicted label. This can be explained by the fact that the decision, even if it resulted in the same class assignment, was based on two different feature sets. Another interesting finding of the authors is that the similarity between models and explainers is negatively correlated with length, but positively correlated with the type-token ratio.

In the final experiment, the authors examined the distribution of important features obtained from the different approaches and used an entropy measure to understand how chaotic the distribution is. The results show that important features have higher entropy when using LSTM with attention and lower entropy when using XGBoost. In conclusion, the authors note that different approaches can sometimes lead to very different important features, nonetheless, they may bring to the same classification output. There are some consistent patterns between models and methods: deep learning models tend to produce similar important features, independently by the used explainer, and these important features are

different from the ones generated by traditional models such as XGBoost and SVM. Furthermore, the explanation provided by post-hoc explainers are intrinsically similar to each other, and the same similarity can be found within the explainers that are considered built-in.

Lertvittayakumjorn et al. [17] replicated the same approach in another paper, where they used the help of humans in three different tasks to validate the explanation generated by post-hoc explainers. For this purpose, human annotations have been obtained from the Amazon Mechanical Turks, which is an internet service for the automation of certain tasks that computers cannot perform. The first task was the investigation of the model behavior. Humans have been asked which explanation is more reasonable, after showing the highlighted texts extracted from the explainer applied to two different models. The authors noted that humans prefer explanations with more evidence texts and that are more grammatically correct. The second task concerned the justification of certain model predictions, and humans have been asked whether the highlighted texts can be considered related to one class and to which degree they were distinguishable from the other class. The results showed that words cannot be attributed to a certain class without providing a large context. The third task concerned the investigation of uncertain predictions, and humans have been asked to perform the classification in place of the model for all the cases where the confidence was low. It has been shown that humans can be as undecided as algorithms when an adequate context is not provided.

The comparison among the explainers reflected the findings of other papers, and the results do not contradict the ones obtained by Vivian Lai et al.

4.2 Attention as explanation

Attention mechanism offers a natural way to interpret the result, since each token ongoing the classification process receives a weight coming from attention as illustrated in the previous chapters, many people used the attention weights as a rank of importance of tokens, and thus for an overall explanation of the classification. Furthermore, the attention weights follow a statistical distribution, and this could appear like a relative importance measure.

This supposition was undermined in 2019, in a paper published by Wallace et al. and that could be summarised in its own name "Attention is not explanation". In this paper, the authors attempts to demonstrate that there is not a correlation between attention and explanation through experiments. The experiments are performed across a variety of NLP tasks, and try to explain how other feature importance measures correlate with attention weights and if there are adversarial distributions of attention that could lead to the same result. In the first experiment

the authors assess if the learned attention weights agree or disagree with alternative natural measures of feature importance, such as gradient attribution and Leave One Out. Thus, two opposite tests are prepared to evaluate the statistical correlations.

A first result is that gradient attribution and LOO do correlate with each other and, specifically, correlation has been measured between attention and gradient-based measures. Instead, when tested if they correlate with attention weights, the observed correlation is modest. Gradients for inputs in other models, like the embedding based ones, show a higher correlation with attention weights. The hypothesis proposed by the authors is that *BiRNN* induces attention on the basis of hidden states, which are influenced by all the words and then the score is presented in reference to the original input. In conclusion, the experiment results demonstrated that attention weights do not strongly or consistently agree with the considered feature importance measures, with the only exception given by the models that are too simple and thus attention plays an essential role. If we consider the explanation methods as valid, then attention weights could not be considered as valid.

In the second experiment is assessed the existence of an adversarial attention weights distribution that leads to the same prediction. This experiment is justified by the assumption that, if attention weights are explanatory, a hypothetical adversarial configuration may be considered as another potential explanation. If the new distribution does not change the model output, it is then legit to claim that the attention weights are not enough to be considered explanatory for the whole model.

Thus, the authors designed an algorithm able to construct a counterfactual distribution of attention weights. The algorithm works thanks to the adversarial principle, where there is a function to minimise taking in account a constraint to maximise. In this case, the function to minimise is the difference between the output base untouched model and the output of the training algorithm, while the constraint is given by the fact that the attention weights distribution must be as different as possible as the base model.

The conjecture is that, if attention weights are to be considered as explanatory, then if we remove the weights from those few tokens that have the most of it, the prediction should drop considerably and eventually flip. However, this was not confirmed by experiments, in contrast to an intuitive property of explanations: shifting model attention to very different input features should yield corresponding changes in the output. In all the datasets considered by the authors, there existed an alternative attention configuration over inputs that yield the same output, independently by how high was that attention weight originally. The authors provided different evidences that there is not a correlation between intuitive feature importance measures and learned attention weights, and found out that a different distribution of attention weights could lead to the same result. Thus, the ability of attention weights to provide transparency or meaningful explanations for model

predictions could be questionable. In the same year that the paper "Attention is not explanation" [18] was published, Wiegreffe et al. proposed a new paper that contested the findings of Wallace at al. and was therefore named "Attention is not **not** explanation."

Wieggreffe et al. [19] claimed that Wallace at al. conducted the experiments in an environment that left a large degree of freedom, so there is no real way for users to understand whether and to what extent attentional weights are used by the model. The authors propose a more model-driven approach in which the attention weights are first fixed to a uniform distribution in order to actually and preemptively check whether or not they are being used for prediction. Indeed, the researchers cannot assume that the attention weights are related to the explanation if they are not used from the beginning.

The authors demonstrated that the experiments conducted by Wallace et al. were not able to prove their thesis for anything concerning counterfactual attention weights, since it is not found in any part of the definition that attention weights are a primitive, so the alternative distributions found may lead to similar predictions, but since they were not trained on the model, they lack the explanatory power that researchers might expect. Therefore, they propose a new algorithm that takes this consideration into account. In addition, the authors counter the assumption that existence implies exclusivity; in other words, the authors contend that attention weights might be useful in showing **an** explanation rather than **the** explanation. For these reasons, Wallace et al. have not demonstrated that there is an adversarial distribution that leads to the same output, and this model cannot be considered plausible or a faithful explanation. Also, the experiments have not provided any indication of the expected amount of variation in adversarial attention distribution of attention and have not provided any indication of how much adversarial the distribution is.

Accordingly, the authors proposed a careful methodological approach that replicated the experiments but took into account all the weaknesses of the previous approach to avoid them and kept the properties of the attentional distribution in mind. The first step was to define the scope of the model's performance and variance and only then was an empirical diagnostic technique developed and implemented. The diagnostic technique was developed to measure the usefulness of the attentional weights in a model-agnostic fashion, by capturing the relationship between input and output. In a first experiment, the authors fixed the attention weights to a uniform distribution, thus de facto eliminating the attention mechanism. A first surprising result was that predictions did not drop as expected, but remained unchanged for three of the tasks considered. Indeed, in attention-based models, other mechanisms are involved to achieve prediction. If attention is not used for prediction, any attempt to use it for explanation makes no sense. In a second experiment, the authors tested whether or not the variances observed by Wallace

et al. between the baseline model with attention weights found by fine-tuning and the other model found by adversarial training were unusual. This was done by trying different seeds in the adversarial configuration to introduce additional variance. In another experiment, the authors proposed to evaluate the importance of tokens by measuring the backward-pass gradient flow, and introduced a post hoc training protocol of a context-free model driven by given weight distributions for this purpose. This aims to replicate what the authors of the previous work did, but in this case, the predictive power of attention distribution is evaluated in a clean environment where the trained parts of the model do not have access to the next or previous tokens of the considered sample.

This setting led to an important result for all datasets considered, namely that using the pre-trained attention weights is better than the weights obtained by training, which in turn is better than the setting without weights. Thus, since this model performs better than the model trained with uniform attention weights, it is reasonable to assume that attention mechanisms are more important than the underlying word-level architecture, at least for the datasets considered. These results can be seen as evidence against the claim that attention weights are arbitrary and therefore provide no explanation. Indeed, independent token-level models that do not have access to contextual information find the attention mechanism useful, indicating that at least these models encode some measure of token importance.

4.3 Recurrent patterns in attention maps

In a famous paper named "Revealing the BERT dark secrets", Olga Kovaleva at al. [20] perform an analysis of the patterns that attention maps adopt after fine-tuning a BERT model. The work was completed by paying special attention to the interpretation of self-attention, which is considered the most important underlying component of BERT. An initial finding is that there are a limited number of attention patterns that are repeated across the heads. As a result, not all the attention weights in the attentional maps are equally important, so the model is overparameterized. In addition, although the different heads adopt the same pattern, they have different effects on performance in each taken task. Nevertheless, BERT is the de facto model for all NLP tasks, the exact mechanism that made its performance so good remains still unclear. The authors of the paper attempt to address this problem by defining a methodology and conducting some experiments that analyse the patterns that attention maps adopt in BERT after they have been fine-tuned. The experiments aim to capture different types of linguistic information by encoding them in self-attention weights. Furthermore, the authors present evidence that BERT is over-parameterized. They show a gain of up to 3.2% when the number of weights is reduced. This hypothesis is supported

by the fact that in 2019 Jawahar et al. showed that the intermediate layers of BERT capture important linguistic information, and Liu et al. showed that the middle layers of Transformed-based architectures are the most transferable to other tasks. In the same year, Voita et al. demonstrated that only a few of the 144 heads are useful for the translation task. The authors use this foundation to understand an important issue, namely whether there are common patterns of attention, how they change during fine-tuning, and how this affects performance.

To answer these questions, the authors have proposed 4 experiments, all of which are relevant to this thesis but only the first is crucial to the overall understanding. Across all the experiments, the authors extracted from a given input an $L \times L$ matrix previously defined as attention map in the previous chapter of this thesis. These experiments aim to analyse how BERT processes different types of linguistic information, syntactic roles, semantic relations, and negation tokens. In the first experiment, the author performed a manual inspection of the self-attention maps and found that there are only a handful of patterns that are repeated. All patterns can be exhaustively divided into these 5 categories:

- Vertical
- Diagonal
- Vertical + Diagonal
- Block
- Heterogeneous

The patterns are not all equally represented. In fact, the heterogeneous pattern was the most frequent, while the vertical pattern accounted for 30% of all samples. The vertical patterns can be explained as a focus on a single token of the sentence, thus the results output of the self-attention mechanism only depends on the presence of that token and not of the presence of all the others. Instead, the diagonal pattern indicates that each token is paying its attention mainly to the previous token in order to make a sense from the sentence. The Vertical + Diagonal patterns are a mixture of both. Block patterns, instead, indicate that the attention is limited to a certain amount of previous or following tokens. Last, the heterogeneous pattern includes all the cases not considered in the classification, and it just indicates the normal working of the self-attention mechanism. The upper bound for the heterogeneous category has been estimated to be between 32% and 61%, depending on the task.

In the second experiment performed by Olga Kovaleva at al., an attempt has been made to understand which syntactic and semantic relations are captured by self-attention patterns. The authors decided to investigate semantic role relations

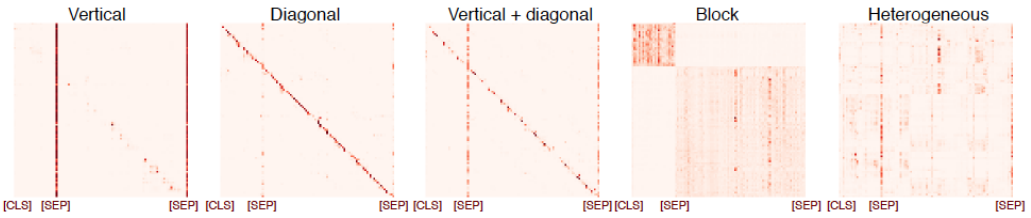

Figure 4.1: Example of the 5 recurrent patterns, taken from the original paper

defined in frame semantics. In this experiment, the heat map of averaged attention weights across all collected examples indicated that 2 out of 144 heads tended to focus on the parts of the sentence that FrameNet annotators identified as core elements of the same frame.

In the third experiment, the effect of fine-tuning on performance was analysed by calculating the cosine similarity between the flattened arrays of attention weights given the pretrained version of BERT and the corresponding fine-tuned version. The authors found that the last two layers encode task-specific features responsible for the gain in scores. The earlier layers are still useful, since the information captured is used in the fine-tuned model. Finally, the fourth and last experiment investigated whether certain linguistic features are emphasized by BERT. The result was negative, as the vertical attention pattern was predominantly assigned to the special tokens *[CLS]* and *[SEP]*. The overall results suggest that even the base BERT model, which has fewer weights than the large model, is effectively over-parameterized. Evidence for this claim is the repetition of self-attention patterns in different heads, together with the fact that disabling most heads has no significant effect on performance, and in some cases actually increases it.

Chapter 5

Overview of explanation techniques

In this chapter different explanation techniques, used to various extents in this thesis, will be illustrated. First of all, a definition of explanation is provided, in order to understand what are the essential properties that explanation must have and what are the reasons why some choices have been made.

Following, LIME and Shapley values will be introduced to have a more complete vision of SHAP, which has been largely used in this thesis.

In the end, two ad-hoc explanations frameworks are defined, namely MiCE and SOC.

5.1 Explanation in Machine Learning

Having a high-performing model is not considered enough if is not possible to answer other questions, such as if the model is trustable if it will perform well also after it has been deployed, and what else it can tell us about the world. For this reason, the model has also to be interpretable.

Lipton gives a clear definition of interpretability since it was not previously well defined in other works, nor it was explained why it was assumed important, despite being used as a solution to many of the problems listed above.

Interpretability, as suggested by Lipton, is not a monolithic concept, but instead, it reflects different distinct ideas, such as trust, and causality. In this work is only considered the explanation of supervised learning models, and are not considered other ML tasks, such as unsupervised learning and reinforcement learning.

Furthermore, even if many papers define interpretability as a mean to reach trust [21]), this only redirects the problem since now the concept itself of trust is needing of a definition. In addition to this, in recent years many legal institutes included

the European one, introduced the legal requirement of the right to explanation and thus requiring interpretability in sensible processes. The desire for an interpretation suggests that predictions and metrics calculated on these predictions are not enough to describe the model. A work has been performed by Lipton and resulted in 5 desiderata that a model must contain:

- **Trust**, that is the amount a model can be trusted, measured as the times a model is right and for which examples it is right. A trustworthy model is a model that tends to make the same mistakes that a human makes, and that is accurate when a human is accurate. A sufficiently accurate model is not automatically trustworthy.

- **Causality**, that is the ability of a model to show causation, the model should be able to make associations, and properties should be inferred for generating hypotheses about the natural world.

- **Transferability**, the model must be able to generalize in unseen scenarios, and the distribution of the population seen during the training must be the same that will see after the deployment. If the model is deployed in a setting where the user might alter the environment, their future predictions are thus invalidated.

- **Informativeness**, even if a model minimized a specific error, this does not mean that it will provide real-world useful information. An informative model is a model that can be used to make decisions. A model can be informative even if its inner working remains obscure.

- **Fairness** the model should provide the information to assess if the decision produced automatically conforms to ethical standards

In her paper, Cynhia Rudin defines explainability as correlated with the decision-making process [22], and it argues that interpretability, while more desirable, is more difficult to achieve than explainability because it must provide people with a comprehensive understanding of the correlative relationship between inputs and outputs.

In another paper, Lei et al. [23] explain how cannot exist a prediction without the justification of the reasons why that prediction has been made, and if it exists its applicability is then limited. As a remedy, they propose to train a neural network to extract pieces of input text as justifications, in the paper named rationales, that have the constraint of being short and coherent, yet sufficient for making the same prediction. The proposed approach combines two modular components, generator, and encoder, which are trained simultaneously to generate rationales

and predictions from the input text, and gold-label rationales are used to evaluate the model.

Thus, the evaluation has been accomplished by comparing the result against manually annotated test cases. This approach is called extractive.

5.2 Explanation taxonomy

The existing explanation methods used for classifiers can be divided into homogeneous groups, according to their characteristics. [24]

First of all, an explanation can be **intrinsic** or **post-hoc**. An explanation is considered intrinsic whether an interpretation is reached by applying changes to the classifier that result in a simplified version of it. An example could be in a neural network in which, in order to get an interpretation out of it, a linear regressor is trained to behave as similarly as possible to it. On the other hand, an explanation is considered post-hoc if the analysis is performed on the trained classifier, by perturbing the input and observing the variation in the output. An example of post-hoc interpretation is the permutation feature importance.

There are other two families which classifiers can be divided into, and they are **model-specific** and **model-agnostic**.

The family of model-specific interpretation tools includes all the tools that are designed on purpose to work exclusively on that specific model, as the interpretation of regression weights in a linear model which assumes meaning only in the context of regression. Also, tools that were designed to interpret only neural networks are model-specific. Instead, model-agnostic tools can be used independently by the classifier. Usually, these tools only are post-hoc and only rely on the output and on the input features.

Lastly, an explanation may be **local** or **global**. An explanation is considered local whether it only explains a specific prediction and not the whole classifier, while it is considered global if it provides insights about the whole model.

Furthermore, the classification methods can be differentiated according to their results:

- **Feature summary statistic**, if a statistic for each feature is provided
- **Feature summary visualization**, if dependencies are graphically visualized
- **Model internals**, if details about the internal working of the classifier are included in the output
- **Data point**, if data points are generated to make the model interpretable

- **Intrinsically interpretable model**, if the model is so simple that does not need any interpretation but instead can be used as a means to interpret other complex models.

5.3 LIME

LIME is an acronym that stands for "Local Interpretable Model-agnostic Explanations" [25], and it is an explanation technique that aims to explain the output of a classifier by learning an interpretable model around the prediction, using a local approach.

The training of the interpretable model also called the surrogate model, is reached by selecting an instance of interest from the dataset, which an explanation needs to be provided. The the dataset is perturbed and the classifier gets fed with the newly generated data to obtain a prediction. The new samples are thus weighted accordingly to their proximity to the instance of interest. Thus, the surrogate model is trained on the dataset with the variation, and the explanation is reached by analysing the interpretable model.

5.4 Shapley values

The Shapley values were introduced in the cooperative game theory context, by Nodel Prize Lloyd Shapley, which it takes the name from.

It aims to assign a unique distribution for each cooperative game, from a total surplus generated by the coalition of all players. In other words, Shapley values satisfy the property of fair distribution of gains among different players, considering the case where they collaborate. Players will follow the external enforcement of cooperative behavior.

Shapley values are used as a base for the implementation of SHAP.

5.5 SHAP

Explanation is a complex tasks, and many methods have been proposed across the time to let users interpret the predictions of obscure models. However, understanding how these methods are related with the task and when the interpretation is reliable raises new concerns. For this reason has been introduced **SHAP** [26], which stands for SHapley Additive exPlanation, and aims to create a new unified framework for interpreting predictions.

In practical terms, SHAP calculates the contribution of each feature to the prediction of an instance x to achieve the explanation of a model. Thus, in a

prediction is assigned an importance value for each feature, to rank them and obtain an explanation of which features are playing a more important role.

The Shapley values are calculated using the SHAP explanation technique, which is based on coalitional game theory. The Shapley value explanation is depicted as an additive feature attribution approach, a linear model that is one of the innovations of SHAP. SHAP specifies the explanation as:

$$g\left(z'\right) = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{5.1}$$

To compute the Shapley values, we simulate that only some feature values play a role (named as *present*) and some do not (named as *absent*). For x, the instance of interest, the coalition vector x' is a vector with all ones, i.e., all feature values are *present*. Thus, the formula simplifies to:

$$g\left(x'\right) = \phi_0 + \sum_{j=1}^{M} \phi_j \tag{5.2}$$

Another result is that the only solution that satisfies the properties of efficiency, symmetry, dummy, and additively is Shapley values. SHAP claims to have three additional desirable properties, and they are local accuracy, missingness and consistency.

By **local accuracy** is meant that the learned model should be a good approximation of the machine learning model predictions locally, and this can be formally described with the following equation:

$$\hat{f}(x) = g\left(x'\right) = \phi_0 + \sum_{j=1}^{M} \phi_j x'_j \tag{5.3}$$

Instead, the **missingness** is a property obtained when the missing features get an attribution of zero, that is something that we intuitively expect but is not for granted. The missingness property can be formalized in the following way:

$$x'_j = 0 \Rightarrow \phi_j = 0 \tag{5.4}$$

Finally, the **consistency** property is obtained when the explanation model reacts to changes in a way that the marginal contribution of a feature value increases or stays the same regardless of other features by keeping the Shapley values the same or increasing as well. In other words, if the marginal contribution of a feature increases the corresponding Shapley value will not decrease. Also this property can be expressed with an equation:

Let $\hat{f}_x\left(z'\right) = \hat{f}\left(h_x\left(z'\right)\right)$ and z'_{jj} indicate that $z'_j = 0$. For any two models f and f ' that satisfy:

$$\hat{f}'_x(z') - \hat{f}'_x\left(z'_{\setminus j}\right) \geq \hat{f}_x(z') - \hat{f}_x\left(z'_{\setminus j}\right) \tag{5.5}$$

for all inputs $z' \in \{0,1\}^M$, then:

$$\phi_j\left(\hat{f}', x\right) \geq \phi_j(\hat{f}, x) \tag{5.6}$$

The SHAP theorem states that there is only one solution that satisfies all the properties previously mentioned, and it is the solution of the equation:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!} \left[f_x(z') - f_x(z' \setminus i)\right] \tag{5.7}$$

where $\phi_i(f, x)$ are the Shapley values related to the classifier f and the feature x.

The SHAP theorem is a result of combined cooperative game theory, the understanding and the demonstration are out of the scope of this thesis. In 1985, Young et al. demonstrated that Shapley values are the only set of values that can actually satisfy at the same time the local accuracy and consistency properties, while the missingness property is required to transfer the obtained result to the class of additive feature attribution methods. Since the implication is valid in both the direction, another result is that all the methods that are not based on Shapley values may violate local accuracy, consistency or both. However, SHAP values could be a problem because of their high computational cost. A solution can be found using some heuristic that become valid using the additive properties. The authors propose a novel method to calculate Shapley values using kernel, that is covered under the name of **KernelSHAP**.

KernelSHAP is an alternative kernel-based estimation approach for Shapley values, that has been inspired by local surrogate models.

First of all, a kernel is needed, and this can be defined as:

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}\,|z'|\,(M - |z'|)} \tag{5.8}$$

After the definition of the kernel, only five steps are required for the calculation of SHAP values:

- Sample a set of features
- For each coalition, get a prediction only based on that
- Compute the weight for each coalition, using the SHAP kernel
- Fit weighted linear model

- Extract the weights from the linear model, that now corresponds to Shapley values

The Kernel works also when many features are involved.

Conversely, **DeepSHAP** is a different flavor of the SHAP algorithm specifically designed for neural networks, to leverage the additional information that can be extracted from the weights and the gradient, and thus have better performance and results. The implementation of DeepSHAP is based on an adaptation of DeepLIFT.

5.6 MiCE

MiCE is another renowned solution for providing explanation, and stands fro Minimal Contrastive Editing [27]. It aims to simulate the way humans have to explain something to themselves, that is imagining a contrastive example and wonder why the contrastive example did not happen while the real example did happen.

A contrastive explanation can be conceptualized as the answer to all questions of the form "Why p and not q?", where p is the even that actually happened while q is the imagined conterfactual event, which is also called the contrast case. MiCE proposes a new method for producing contrastive explanations of a NLP neural network as edits to inputs, as addition, change or removal of tokens, as minimal as possible as they change the model output to contrast case. The edits produced by MiCE are not only contrastive, but also minimal and fluent. Thus they can be used for debugging incorrect models and uncovering unwanted artifacts on a model.

To produce the contrastive explanation, MiCE adopts a two-stage approach. In the first step the editor will be fine-tuned, which is a model implemented by HuggingFace under the name of Text-To-Text-Transfer-Transformer (T5), and as the name says it is an architecture based on attention that aims to learn a pattern from a sequence of tokens to another sequence of tokens, and thus will be used to learn the edits. In the second stage, the editor fine-tunded in the previous stage will be used to actually obtain the edits.

To do this, Mice masks out a consecutive spans of a certain percentage of tokens (where the percentage can change across the run) in the original input, then the contrast prediction is prepended to the masked input, and thus the resulting masked instance is used as an input for the EDITOR.

5.7 SOC

A different approach has been taken by SOC, which stands for Sampling and OCclusion [28], and it aims to explain a neural network by providing a hierarchical structure where set of consecutive tokens are gathered together and classified.

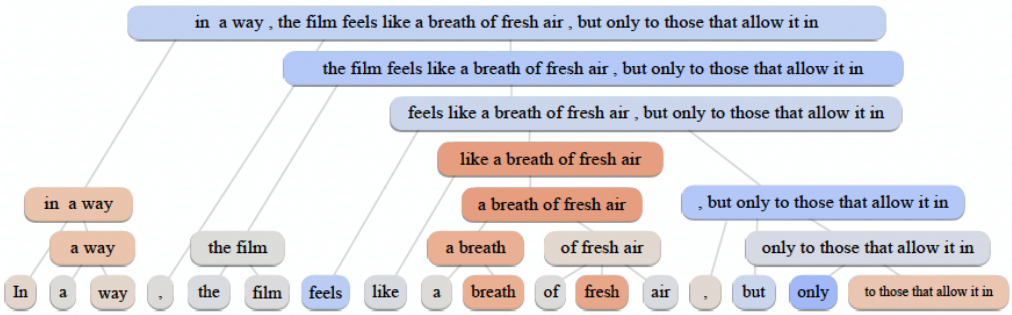

Figure 5.1: Example of hierarchical explanation

The explanation offered by previous models has always been seen as limited since, while they offered an explanation at the single token level, they lacked an overall explanation when those tokens were combined with each other. Thus they are not a valid solution for explaining the importance of a phrase rather than a word, since a phrase importance is often a non-linear combination of the difference importance of the tokens in the phrase. To obtain the phrase-level importance, SOC splits a sentence into more subsets of contiguous tokens, then it identifies the importance by combining two ore more of these subsets and studying how the importance of the combined subsets differs from the importance of the single subsets summed.

The SOC authors introduced a mathematically formulation to formalize and approximate the context independent importance, and is called N-context independent importance, and is defined as the difference of the output of the model after that the subset tokens have been masked out, marginalized over all possible N words surrounding those tokens in the sentence. SOC is simple and model-agnostic, and has demonstrated to provide state of the art performance against previous algorithms. It works in a simple way, it uses an input occlusion algorithm which goal is to calculate the importance of a subset of tokens specific to an input sentence, by observing the prediction difference caused by replacing the subset of tokens with padding tokens.

Obviously any change in the subset of tokens will produce a different production, and in order to remove the dependence SOC uses an approach where the context around the tokens are sampled and randomly replaced. The replacement of the sampled tokens is obtained through a trained language model where a set of neighboring word replacement is extracted. For each replacement, thus the algorithm calculates the model prediction differences after replacing the subset of tokens with the padding tokens. The importance is then calculated as the average prediction differences. The combination of sampling and occlusion are the two

single ingredients of SOC, as the name states. The input occlusion algorithm can also be substituted with other measure of phrase importance, such as Shapley values.

Chapter 6

Methodology

This thesis proposes the objective of exploring the bias in the hate speech context, using a neural network architecture specialised in sentence classification and trained to recognize whether the sentences are toxic or not. A comparison among all the explanation methods is offered, and a correlation between some pattern in the attention weights and the false positive is investigated. In the end, a correlation analysis between the different explanation methods has been performed.

6.1 Approach

Because of the heterogeneous types of tasks treated in this thesis, different approaches have been adopted in each phase.

In the first phase, an analysis among the different explanation methods has been conducted to understand what are methods that offer a better explanation in the context of toxicity detection.

Explanation may be a subjective fact, asking different people to explain a sentence may lead to different and contrasting results. For this reason, even if the analysis may appear subjective to the author of this thesis, I defined some objective parameters to select the best explainer among different methods and kept an objective and impartial approach, to offer a qualitative though objective analysis.

To perform the comparison, the attention was focused only on 10 sentence. From the test dataset 10 misclassified sentences have been extracted, 7 false positive and 3 false negative. The work has been conducted only on misclassified sentences since the aim of this thesis is to analyze and explain the bias, thus in which cases and in which circumstances the neural network takes a certain decision, by highlighting the word or the group of words that have led to the mistake. Furthermore, the misclassified sentences have been selected in such a way that to obtain a 70%

of false-positive and a 30% of false negative. The majority of false-positive is motivated by the fact that, in the hate speech context, a false positive is of greater importance than a false negative.

In case of a false positive, indeed, a user may be flagged as toxic while his or her post has no elements to be classified as that. This could make the experience in the online community worse or impossible to tolerate. Furthermore, from a technical point of view, the research of false-positive might be more interesting to analyze, since few words may lead the whole classification and thus a focus on those few words can be more easily analyzed and eventually adjusted.

The false-negative cases, however, are not to be belittled. These cases involve all the sentences where a misogyny attack has been expressed, but the neural network failed to notice it. Also, a high false-negative rate could conduct to a worse online experience, even if posts containing misogyny are just a minority if compared to the total amount of posts. For this reason, also a certain amount of false-negative have been included in the analysis.

The 10 sentences have been manually selected from the test dataset. For this purpose, some parameters have been defined for the selection, these parameters aim to obtain an as vast as possible range of mistakes. Indeed, even if the sentences are all different, some of them represent the same type of mistake, thus the analysis would be redundant. So, the most important parameter that drove the selection has been the expression of a range of mistakes and not the focus only on one type. Secondly, some sentences have been selected because it was not clear what type of mistake has been made from the neural network. And, finally, some sentences have been selected to cover different syntactical parts of a sentence, e.g. in some sentences, the mistaken part was contained in the direct object, in other sentences, it was contained in the verb.

A more detailed analysis of the criterion used for the selection is proposed in the next chapter, which treats experiments and results.

After that, a comparison among all the explainers has been conducted. The comparison included the ten selected sentences and the different explainers. In the following list, all the explainers are listed and a brief description of the shape of the explanation is provided.

- **Raw attention.** The output is provided as a self-attention map, where the attention of a couple of tokens is represented in a matrix. The explanation is then obtained from the tokens that obtained more or less attention

- **Effective attention.** Similar to the raw attention, but the self-attention map represents the effective attention, that is the attention that is effectively used for the classification.

- **MiCE.** The output is composed of the list of contrastive edits performed on

the sentences to obtain the flip of the classification. For each sentence, a list of ten contrastive edits is provided and they are sorted from the most minimal to the most verbose. From the edits can be obtained the table of which tokens have been added, deleted, or modified.

- **SOC.** The output is graphically represented as a hierarchy. At every level of the hierarchy, a different group of tokens is represented, and the number of tokens that compose a group is different. In the top-most level, is represented the classification of the sentence as a whole. In the bottom-most level, is represented the classification of every token considered separately. The tokens or the group of tokens are colored differently, if they are gray they didn't compete for the class label attribution, more the color tends to red and more they competed for the positive class attribution, and finally, more the color tends to be blue and more they competed for the negative class attribution.

- **DeepSHAP.** A vector of the size of the sentence tokens is obtained, where also the padding tokens are considered. In each element of the vector is represented the SHAP value corresponding to the token. Higher is the SHAP value and higher is its importance for the class label attribution.

- **KernelSHAP.** Similar to DeepSHAP, the only difference is the algorithm used.

- **Hidden Token Attribution.** The Hidden Token Attribution is calculated for the *[CLS]* token in the last layer with respect to each token in the input sentence. The obtained values are then organized in a vector.

After the training of the neural network, each explainer has been applied to explain the test dataset sentences, and the output has been saved in a different folder.

The research has proceeded by comparing the output. The comparison has been conducted by assigning a score to different explainers, and during the analysis, the explainers were considered either independently one from the others or as pairs, in order to highlight strengths and weaknesses. Eventually, the output of the different explainers has also been compared with the human explanation.

Also in this case the comparison among the explainers has followed the definition of a set of parameters, in order to minimize any bias into the decision and keep an impartial approach. The **parameters** used for the comparison are:

- **Quality of the explanation.** The output of the explainer must effectively represent a measure of importance for each word in the sentence. Thus, quality is intended as the way in which an explainer assigns more importance to the most relevant words of a sentence, while lesser importance is assigned to other

words. Generally speaking, an explainer that assigns high importance to the stopwords, or to words that aren't the focus of the sentence but just marginal, is considered a bad explainer. On the other hand, an explainer that assigns a high importance score to the most relevant words, and the importance score is proportional to the effective relevance within the sentence, is then considered a quality explainer. The inner meaning of what is relevant in a sentence and what is not is left to intuition.

- **Ease of reading the result.** The output of the explainer must be easily readable and interpretable, also for not insiders. In case the output is a vector of scores, each score must have a precise meaning. In the case of graphical output, it must be easy to read and interpret, also for people not belonging to the specific field.
- **Background and reputation.** The explainer must have a good reputation. Some explainers enjoy solid theoretical foundations and have been adapted to several specific cases and they are renewed for good performance. On the other hand, many of the methods used in this research to explain the decision of neural networks were not even designed to provide an explanation. Such are the cases of attention-based explainers.
- **Difference with human explanation.** The output must be at least comparable to human explanation. The explanation must ultimately serve humans, for this reason, is expected that a human explanation would perform similarly or better than machine drive explanations, representing an upper-bound limit.

About the last parameter, related but independent research has been conducted concerning the explanation provided by humans. The research has been conducted by asking some volunteers to fill out a questionnaire containing ten questions, one for each sentence selected. For each of the ten sentences, it was required to specify whether it was considered misogyny or not, according to the personal opinion of the interviewed person. Both if the volunteer flagged the sentence as misogyny or not, it was then asked to explain why that decision was made. The volunteer had to provide the explanation by selecting up to three words from the sentence itself, that in their opinion drove most of the information about the decision. This approach is similar, in most cases, to the one adopted by neural networks.

Another experiment has been set up to determine whether exists a correlation between vertical patterns and misclassified sentences, expressed as, separately, false-positive rate and false-negative rate. Since the misclassified sentences are extracted from a dataset that concerns hate speech, thus the false positives represent the sentences that did not contain hate speech but were classified as such, while the false negatives represent the sentences that contain hate speech but the neural network failed to notice it.

Indeed, as already reported in the previous chapters, when analyzing attention maps is possible to observe some recurrent patterns, that repeat with different relative frequencies across different experiments, as stated in other publications. Those patterns can be exhaustively classified into 5 different categories that comprise *vertical*, *block*, *diagonal*, *vertical+diagonal*, *heterogeneous*.

Among all these patterns, there is only one that could be of any interest for the research of bias in neural networks, and that is the vertical pattern. Indeed, a vertical pattern represents a focus towards a single token of a sentence, this may lead the classifier to base the whole classification exclusively upon the token that received the self-attention with respect to all the other tokens. Under this hypothesis, the classification may be misled and it could introduce a bias into the neural network internal working.

To test the hypothesis, an algorithm has been created to detect the vertical patterns and thus extract the tokens that were represented in those patterns. The experiment has been repeated varying the minimum lengths that defined whether a pattern was effectively vertical. The results of this algorithm were then gathered into a graph to better visualize the result. For this part of the research, a quantitative approach has been kept, and the results are offered as numbers or graphical representations of those numbers. Since automation was introduced for the detection and the reporting of the vertical patterns, it has been possible to use the whole test dataset, and the results are thus representative of the whole used dataset.

Following the research presented in the first part of this chapter, an additional analysis has been conducted regarding the used explainers, to improve and better understand why some results have been achieved. The analysis involves the research of a correlation among all the used methods, and it has been conducted under a quantitative point of view. The result is composed of the correlation score across all the used explainers, if a correlation between two or more explainers does exist, then is legit to consider the combined use of those explainers as redundant, since the use of one can partially or completely replace the other one. The correlation analysis involved the creation of a matrix that contained the tokens of the concatenated sentences on the horizontal ax, and the different explainers on the vertical ax.

The matrix has been filled with a *1* whether the explainer considered that token important for the determination of the positive class, with a *0* whether the explainer considered that token not important for the determination of the positive class, and with a *-0.25* whether the explainer considered the token as competing against the determination of the positive class, in the eventuality the explainer supported this case. A problem occurred because the fact that many explainers work at word-level while other explainers work at token-level. For this reason, a different approach has been taken in the first case. Indeed, in the case the explainer works at word-level, the assigned score has been repeated for each token composing

that word.

After the filling of all the fields composing the matrix, the Pearson correlation coefficient has been calculated between couples of explainers, composing thus a correlation matrix. The experiments have been conducted on a single dataset, but they may be extended to different datasets if needed.

Chapter 7

Experiments and Results

7.1 Dataset

The first used dataset for this research was taken from the Automatic Misogyny Identification contest, a project founded by the Università degli Studi di Milano-Bicocca, collaborating with the Universitat Politècnica de Valencia. The dataset is composed of three different CSV files, namely the train set, the test set and the dev set. The latter was not used for this research. The corpora that composed the dataset had been manually labeled by several annotators to express the following pieces of information:

- Misogyny or not misogyny
- Misogynistic category, in case it was labeled as misogyny
- Target, that may be passive or aggressive

Among all these components of the dataset, the only used part was the one concerning the fact whether a sentence is misogyny or not. The dataset consists of 3600 train sentences, 1000 test sentences, and 400 dev sentences, for a sum of 5000 total sentences. The second used dataset concerned the hate speech against immigrants. Similarly to the misogyny dataset, this dataset is in the English language and is composed by a collection of tweets, most of them involving the USA society during the Trump election. Indeed, most of the tweets regard hate or intolerance against Mexicans, black Americans, and other minorities present in the USA society, and the figure of Donald Trump is recurring in the posts. Also this dataset was already split into train, dev, and test dataset, and the dev test was not used across the experiments. All the splits contain 40% of positive samples and 60% of negative samples, resulting in slightly imbalanced. Some techniques of rebalancing were applied for the training and the calculation of the test accuracy,

especially through undersampling of the majority class, since the nature of the dataset did not allow any other technique.

7.2 Training

For this thesis, only one implementation of BERT has been used across all the experiments, and a single training for each dataset has been performed and re-used in all the phases. The implementation used was *BertForSequenceClassification* from the *HuggingFace* library, which was previously trained on a large corpus of English sentences as described in the BERT paper. Following the rules for BERT fine-tuning, the neural network has been fine-tuned for 4 additional epochs using the AMI dataset, exclusively basing the classification on the misogyny category.

The following parameters have been used for the training on the misogyny dataset:

- **Epochs:** 4
- **Batch size:** 32
- **Optimizer:** AdamW
- **Learning rate:** 2×10^{-5}
- **Epsilon:** 10^{-8}

Some of the parameters, such as the learning rate and the number of epochs, have been decided after some experiments, in order to maximize the accuracy of the classifiers. The fine-tuning has been executed using exclusively the training set, and it has reached an accuracy of *72.32%* on the test set for the misogyny dataset. Conversely, the training on hate speech against immigrants reserved more problems and required more attempts. Indeed, this dataset contains a larger variety of sentences, and often the distinction between an opinion and a hate speech post is feeble and not immediately recognizable even by humans.

However, after a large number of attempts, the optimal configuration of the hyperparameters is the following:

- **Epochs:** 3
- **Batch size:** 32
- **Optimizer:** AdamW
- **Learning rate:** 4×10^{-5}

- **Epsilon:** 10^{-9}

For this dataset, the accuracy attested to 59% for the whole dataset. However, the dataset is slightly imbalanced, thus the accuracy drops to 54% when considered a balanced undersample of the negative class.

The accuracy is not satisfactory for the use in production of the classifier, but it's anyway useful for the extraction of insights from the neural network inner working. However, the lower number of false-positive samples in this dataset made the misogyny dataset preferential for the analysis performed.

7.3 Selection of sentences

In this section, the list of selected sentences has been reported, altogether with the criterion used for their selection.

The sentences not correctly classified were split into false positives and false negatives in a file, and thus they were manually analyzed and the most interesting was extracted. Between the two used datasets, 15 sentences were selected, trying to keep the proportion of 70% of false-positive samples and 30% of false-negative samples. Indeed, the two types of mistakes are not equally evaluated in their importance, but the false-positive samples are considered to have a bigger impact on a community and they are more strictly connected to the biases assumed by neural networks.

Ten sentences were selected for the misogyny dataset, and they are:

- **Fuck yeah you pussy boy** *False positive* This sentence has been selected because it contains both the word *pussy*, which is a common insult against a woman and the word *boy*, which changes the meaning of the previous insult redirecting it to a boy.
- **@minimaslany What if she enjoys it? I think the days where women are forced to stay in the kitchen, not work, etc are over.** *False positive* This sentence has been selected because it contains a rhetorical question, which may mislead the neural networks, and uses the words "the days are over" to express the contrariety to a concept.
- **@benshapiro Fuck you pussy** *False positive* This sentence has been selected because, even if it is short and similar to the first sentence of the list, it could appear as misogyny at the beginning, from the tag is possible observing that the tweet is referring to a guy, thus cannot be considered misogyny.
- **Why do y'all blame the woman for what she wears when it comes to rape? That shit is WRONG.** *False positive* This sentence has been

selected because it contains a question and it expresses contrariety about the concept using the words "is wrong".

- **Nah yall stay in the kitchen and make bariis and baasto** *False negative* This sentence has been selected because the neural network failed to classify it as misogyny, despite the fact that the author is telling to the girls to stay in the kitchen, which is considered an act of misogyny.
- **If u don't think rape culture exists in the US u should prob check your feed right now 'cause literally every woman u know is posting #MeToo** *False positive* This sentence has been selected because it expresses the contrariety to the concept by inviting people to check that rape culture exists.
- **You don't have to explain why because a dishwasher the right to speak. Stay in the kitchen** *False negative* This sentence has been selected because the neural network failed to consider misogyny sentences as "stay in the kitchen" or appealing a woman "dishwasher".
- **Back to the kitchen you stupid sandwich maker** *False negative* This sentence was selected because the author called a woman "stupid sandwich maker" and the neural network failed to see it as aggressive.
- **Dont call them hysterical women.** *False positive* This sentence has been selected because it's short and simple, nevertheless, the neural network considered it hate speech basing the decision only on the tokens "hysterical women", ignoring the contradiction "Don't call".
- **To the bitch who said the United States is rape FREE. Ask the millions of women who haven't reported it due to how bad our society is.** *False positive* This sentence has been selected because it contains sarcasm and because even if it contains words such as "bitch", it's overall not commonly misogyny.

Instead, for the second dataset, the selected sentences are:

- **A must read article. #BuildThatWall #AmericaFirsty** *False positive* This sentence has been selected because it does not contain any controversial word, with the two only exceptions of the hashtags.
- **Chain Migration is not sustainable and needs to end NOW. It's time to break the chain! #EndChainMigration** *False positive* This sentence has been selected because it expresses an opinion, and it does not contain elements of hate speech.

- **@whaas3 @judithineuropa Just got on twitter because of this farce today. Imagine this, I make a report on You. Calling you names and telling people how big liar and asshole you are without reason. Would you be angry? Of course You would be. Its a same with me. You can fuck off to help some rapefugee** *False negative* This sentence has been selected because it implicitly accused immigrants of being rapists.

- **@CraigRSawyer In ceuta Spain at the borderRefugees go home** *False positive* This sentence has been selected because it does not contain any toxic element, but it was classified as hate speech.

- **The Hillary server/unmasking of U.S. citizens/FBI-CIA-DOJ conspiracy, perjury & obstruction - A SILENT COUP - to take down a U.S. President; by far the largest & deepest scandal to ever hit U.S. government.#MAGA #KAG #Trump #POTUS #NEWS #USA #WalkAway #DrainTheSwamp #NoDACA** *False negative* This sentence has been selected because it spreads fakes news against immigrants.

7.4 Comparison of explainers

7.4.1 Human explainer

A parallel research has been conducted to investigate how humans interpret the selected sentences and eventually use the results as a benchmark for evaluating the quality of the explainers. Indeed, the explanation provided by the different methods should be of any meaning for a human. Thus, the differences between the explanations provided by a human and the one provided by an explainer algorithm should be as similar as possible to consider the latter a quality explainer.

Another reason that drove the decision of investigating the human explanation has been the attempt of removing subjectivity from the analysis performed so far.

The analysis and the comparison among explainers must be as objective as possible, and an impartial and fair approach has been kept across all the experiments. However, all the people are affected by biases, including who is writing this thesis, and this aspect cannot be ignored for a correct classification of the explainers. As an example, one could be inclined to prefer an explainer rather than another, because of some past prejudice that made appear one solution better than another and could transfer this prejudice to the final decision.

On the other hand, asking a list of people external from the experiments what are is the explanation that suits the most a particular sentence could lead to an unbiased and fairer decision, free from the subjectivity that a single inside individual internal to the experiments could bring. When the results are aggregated, the

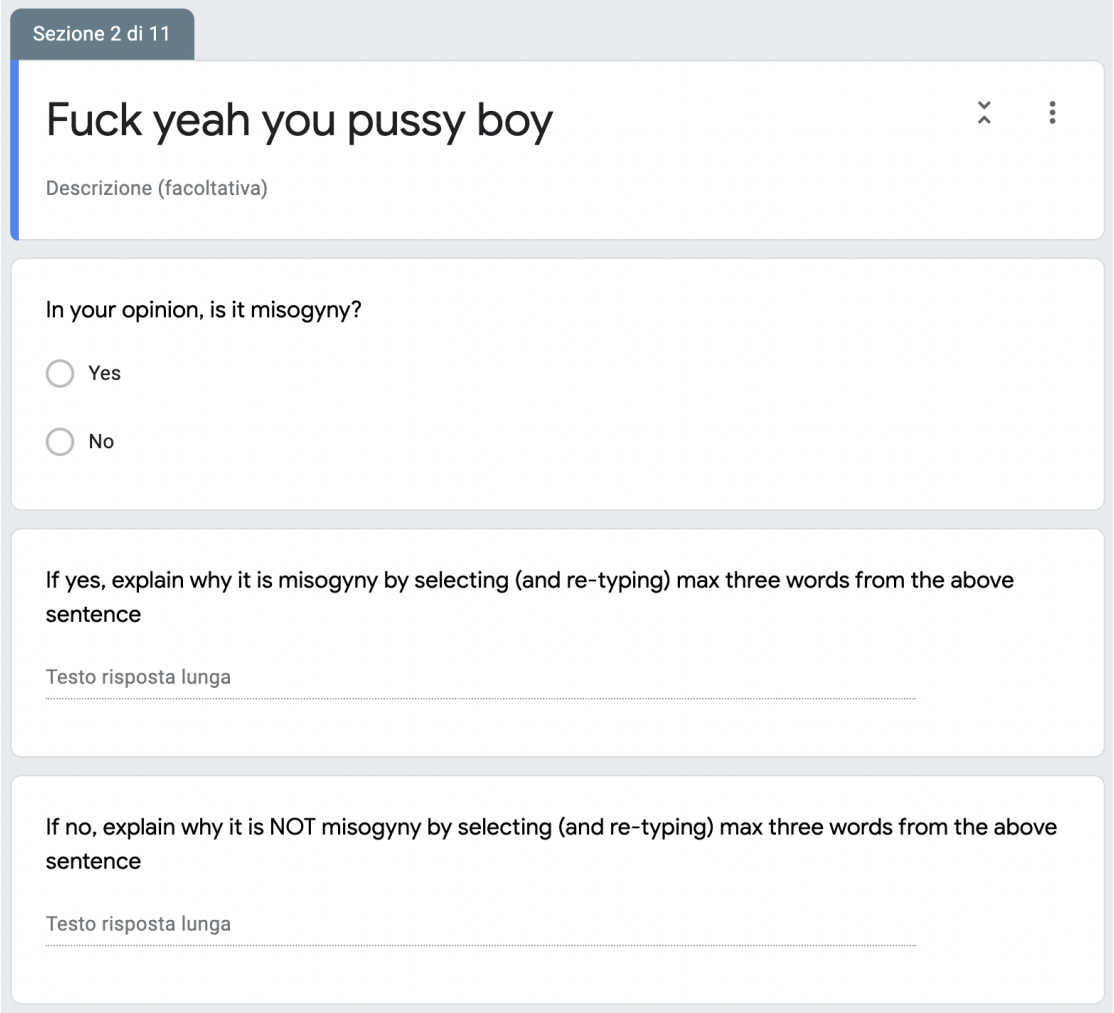

Figure 7.1: Example question of the survey

overall explanation can be seen as shared by a group of individuals. The interviewed individuals were explained how to provide an explanation, but not the finality of the experiment. The interview involved 25 people, and an attempt was made to various age and social backgrounds as much as possible.

The research has been conducted using *Google Forms*, and it is composed by 30 questions, 3 for each of the ten selected sentences.

For each sentence was asked if, in the opinion of the individual responding, the given sentence was misogyny or not. This question was asked to correctly frame the explanation, and to avoid mixing up an explanation of misogyny context with explanations of the not misogyny context.

In some cases, the aggregated answers to this question offered some surprise. As

in the example visible in the figure7.2, where the answer to the question of whether the sentence is misogyny or not misogyny is controversial since almost half of the interviewed individuals did consider it misogyny regardless of the fact that the sentence contained the word "boy". This is understandable considering that "pussy boy" can be also appealed to a woman.

A similar and more controversial example is visible in the figure7.3, where the sentence "@benshapiro Fuck you pussy" is considered not misogyny by 68% of responders. This is probably due to the fact that many of the interviewed failed to individuate the fore tag, which contains the name "Ben". The reason why so many people failed to notice it can be individuated in a lack of focus and scarce knowledge of Ben Shapiro, who belongs to the American culture. The second and the third question are mutually exclusive and depend on the answer provided to the first question.

Indeed, if the interviewed individual considered the provided sentence as misogyny, he or she was asked to select up to three words from the sentence that drove his or her mind towards that decision. The reason why a limit of three words was required was to prevent the responders to select a way too large set of words from the sentence, that would thus invalidate the scope of explanation, and to force the responder to focus on the words that really matter. Similarly, if the interviewed individual considered the provided sentence as not misogyny, he or she was then asked again to select up to three words from the sentence to explain why it was not misogyny. The example results have been then transposed to a *Google Spreadsheet* file, and the answers were manually cleaned and normalized. An example of raw answers contained in the *Google Spreadsheet* file is contained in the figure7.4, where the answers provided to the question "**@minimaslany What if she enjoys it? I think the days where women are forced to stay in the kitchen, not work, etc are over.**" are shown.

Analyzing the different answers, the responders that answered "Yes" to the question "is it misogyny" focused on the first part of the sentence, which reports "What if she enjoys it?", ignoring that the question was rhetorical. Alternatively, they focused only on the toxic words of the sentence. The responders that answered "No", instead, mainly focused on the words "are over", which reverses the meaning of the whole sentence.

In the figure 7.5 the data has been cleaned and aggregated. For each word of the sentence, in the first row, 1 point has been assigned for each user that reported that word as important to explain why it's not misogyny, while in the second row 1 point has been assigned if the user reported that word as important to explain why it's not misogyny. The difference has been reported in the third row, and the background of the scores has been colored consistently to highlight in a more impacting way the important words for the positive or the negative class. The results obtained from the human explainers have been used in the following sections

to express in a subjective way whether the explanation provided by the neural network is plausible or is not of meaning at all. Thus, the human explainer has been used as a benchmark for the evaluation of the other explanation algorithms.

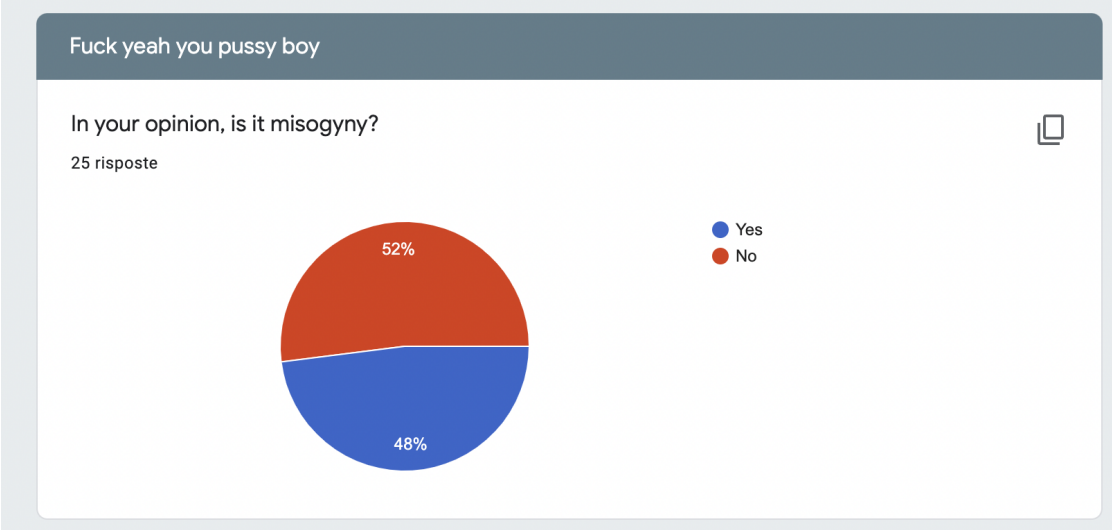

Figure 7.2: Result for the question referring the sentence "Fuck yeah you pussy boy"

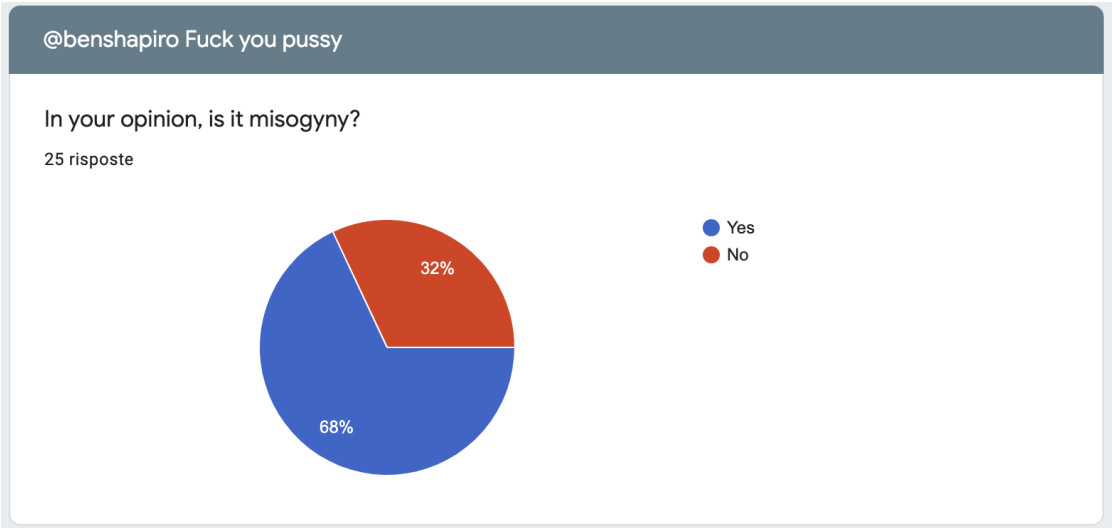

Figure 7.3: Result for the question referring the sentence "@benshapiro Fuck you pussy"

In your opinion, is it misogyny?	If yes, explain why it is	If no, explain why it is NC
No		are over
No		Are over
Yes	If she enjoys it	
No		Forced kitchen over
Yes	Women kitchen work	
No		women forced over
No		
No		Women, kitchen, over
No		over, forced
Yes	she enjoys it	
No		Forced are over
No		kitchen are over
No		are over
No		Days are over
No		Women forced over
No		Are over
No		Over
No		Enjoys
No		days are over
No		
No		days are over
Yes	Forced to stay	

Figure 7.4: Answers to the question "@minimaslany What if she enjoys it? I think the days where women are forced to stay in the kitchen, not work, etc are over."

7.4.2 Attention

Attention is one of the most powerful and basic methods to investigate the internal working of a neural network. By retracing the way a prediction was made, is thus possible to extract information that supported that prediction, which can be ultimately used for an explanation for the classification itself.

To extract the attention maps from the neural network, in the first instance, the sentences are prepared to fit into the neural network. Every sentence undergoes

Word	What	if	she	enjoys	it?	I	think	the	days	where	women	are
Positive	0	1	2	2	2	0	0	0	0	0	1	0
Negative	0	0	0	1	0	0	0	0	3	0	3	0
Sum	0	1	2	1	2	0	0	0	-3	0	-2	0

Word	forced	to	stay	in	the	kitchen,	not	work,	etc	are	over.
Positive	1	1	1	0	0	1	0	1	0	0	0
Negative	4	0	0	0	0	0	0	0	0	10	15
Sum	-3	1	1	0	0	1	0	1	0	-10	-15

Figure 7.5: Aggregated importance given to the words of the sentence "@mini-maslany What if she enjoys it? I think the days where women are forced to stay in the kitchen, not work, etc are over."

a process of tokenization where, using a tokenizer provided by HuggingFace and made on purpose for BERT, the sentence's words are divided into smaller atomic entities named tokens. After that, every list of tokens is filled with *[PAD]* tokens up to the maximum sentences size, which was previously set to 64. Once obtained a consistent list of 64 tokens for each sentence, they are divided into batches, similar of what happened for the training. Each batch is fed into the neural network, with the following optional flag:

```
output_attention = True
```

This asks the BERT implementation to do not discard the attention values after their computation, but to preserve them and produce them as an output of the prediction. As explained in the previous chapters, the self-attention mechanism is a process that relates each token of a sentence against each token of the same sentence, and ultimately assigns a weight accordingly to how much attention the token is paying against the other token. This process is repeated for each head of a layer, and for each layer, BERT is composed of. Since the number of heads contained in each layer is 12 and the number of layers that compose BERT is 12, there are therefore $12 * 12 = 144$ attention maps for each sentence. Given L the number of tokens a sentence is composed of, ignoring the padding tokens, each of the 144 attention maps will have a shape of $L \times L$. The attention values assigned to *[PAD]* tokens are ignored and removed from the graphical representation.

According to Pascual et al., most of the task-specific information of BERT is contained in the last layers, while the first layers encode the information about the language and the syntax [29]. For this reason, the manual analysis and comparison of BERT attention maps focused on the information extracted from the last layers.

What makes attention so powerful is the fact that it is built-in in the neural network, and it analysis and visualization do not need any further processing if not stacking up the outputs into a graph. However, since the high number of attention maps, that cannot be easily visualized in a single glance, and the complex concept of self-attention, this method is considered hard to grasp and lacks many of the

properties a good explainer should have. To stem the high amount of matrices obtained for each sentence, a partial solution was implemented by averaging across the heads for each layer, in this way just 12 attention maps were obtained for each sentence, one for each BERT's layer.

In the figure 7.6 is represented the attention map of the sentence "**Why do y'all blame the woman for what she wears when it comes to rape? That shit is WRONG.**", extracted from the seventh layer and ninth head.

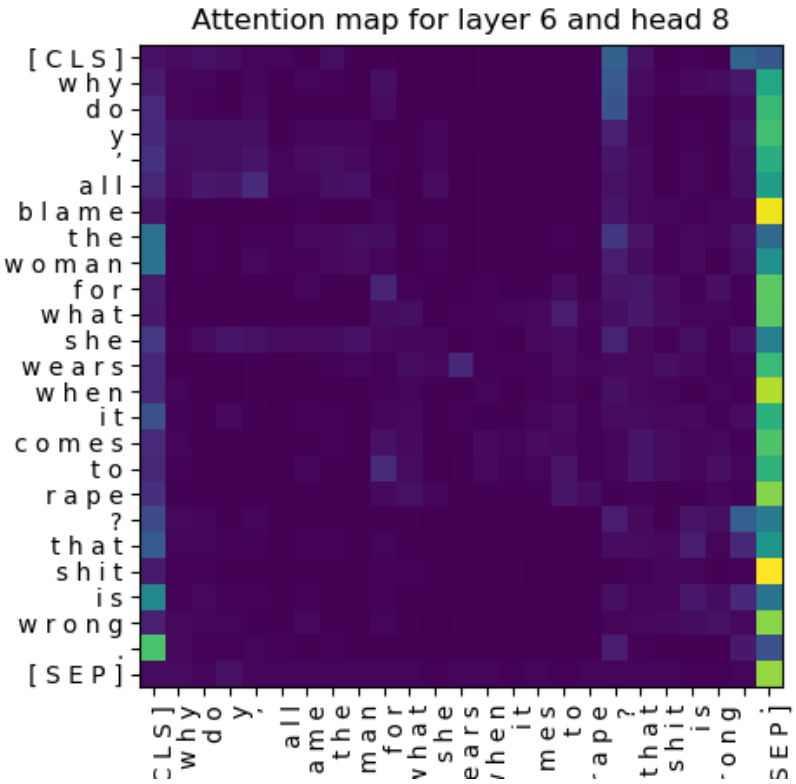

Figure 7.6

as it is visible from the above image, many tokens are paying their attention to the token "rape", which is usually seen in misogyny contexts, while less importance is given to the surrounding tokens, this could explain why the sentence was classified as a false negative. An interesting result is obtained also analyzing the penultimate layer, after averaging the different heads. In this case, multiple vertical patterns come out. This probably is a result explained by the fact that different heads express a vertical pattern over different tokens, and when an average across the heads is calculated, they sum up. The result is reported in the figure 7.7

In this case, it is possible seeing that most of the tokens paid their attention

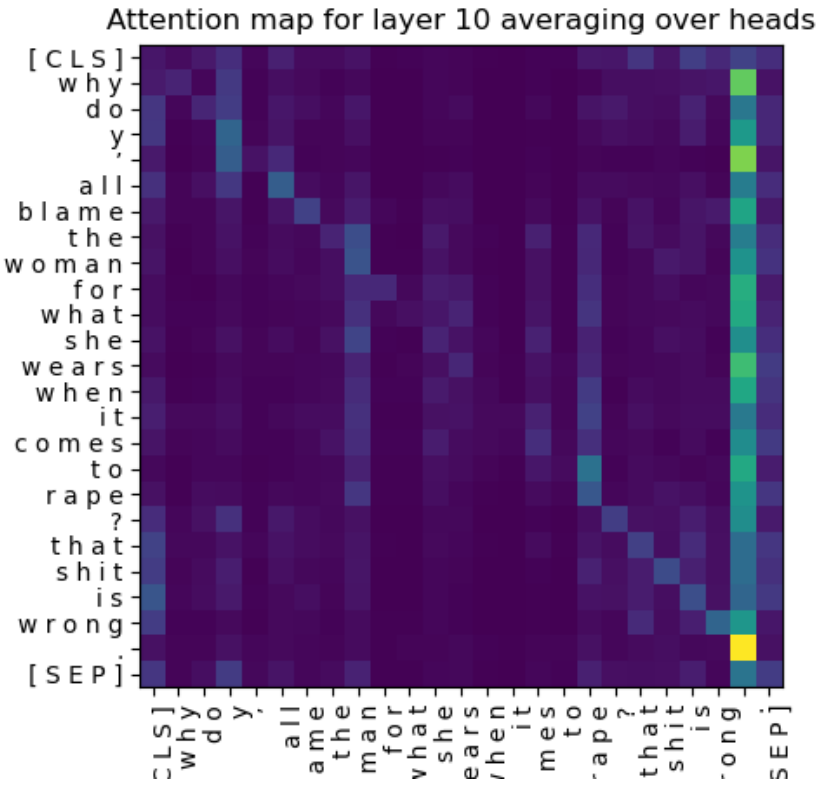

Figure 7.7

exclusively to the tokens "women" and "rape", while a slighter vertical pattern is visible also for the tokens "she", "come" and "do". Most of these words are used in a toxic context and, also in this case, can be used to demonstrate the bias. A similar conclusion can be obtained analyzing a different sentence, that is "**To the bitch who said the United States is rape FREE. Ask the millions of women who haven't reported it due to how bad our society is.**", specifically to the eleventh layer and the average across the heads, as shown in figure 7.8

In the visualization of the above sentence, three clear vertical patterns are visible, and they are related to the tokens "bitch", "rape", and "bad". This shows how the neural networks is ignoring the word "free" that follows "rape", and it is thus missing the overall meaning of the sentence. Furthermore, the neural network is ignoring the presence of the indirect question "ask the millions...", which could have driven the classifier towards a different decision. A different pattern appeared analyzing the sentence "**A must read article. #BuildThatWall #AmericaFirsty**", taken from the dataset that concerns the hate speech against immigrants and is visible in the figure 7.9. Indeed, in the graphical output of the 11th layer and 6th

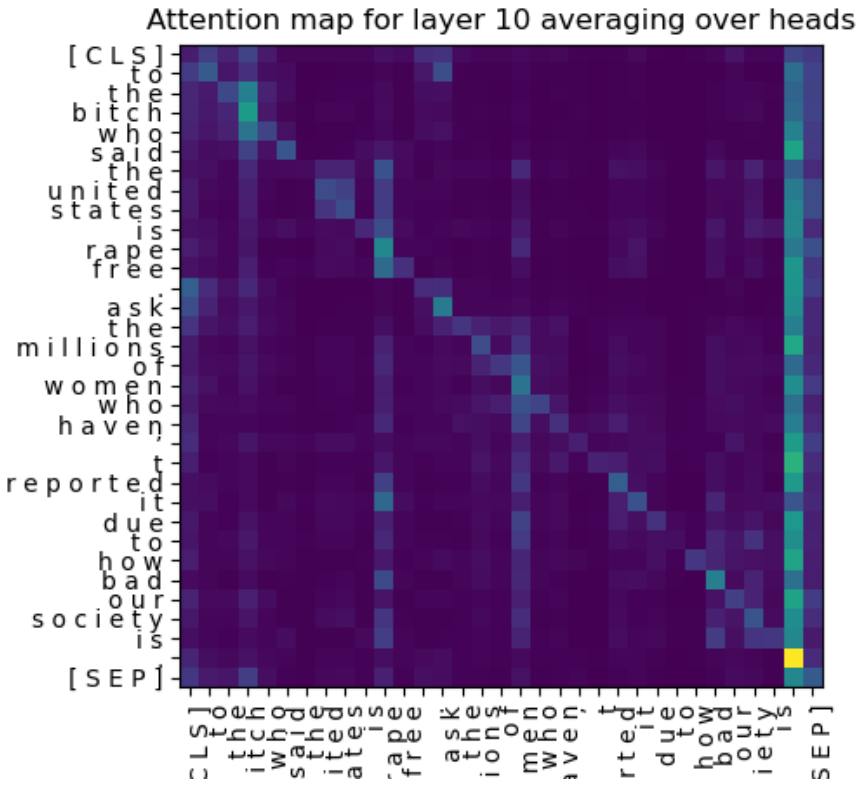

Figure 7.8

head, is clear that there is not a peak around a single token, but the focus regards a set of tokens, particularly the ones composing the hashtags. Thus, the decision was mostly based on the hashtags and not on the content of the sentence itself.

In conclusion, attention maps have proved to be a powerful ally for the detection of bias and debugging of neural networks. Considering that the output of the attention maps is built-in, and it only required the configuration of a flag, together with the fact that its time cost is equivalent to the one required for the inference only, makes attention maps suitable during development time when a rapid insight into the network internal is required for debugging purposes, and a bias can be detected in the early stages of a neural network life cycle. However, their scarce user-friendliness and their bad reputation make them not suitable in most contexts, where a strict explanation is required.

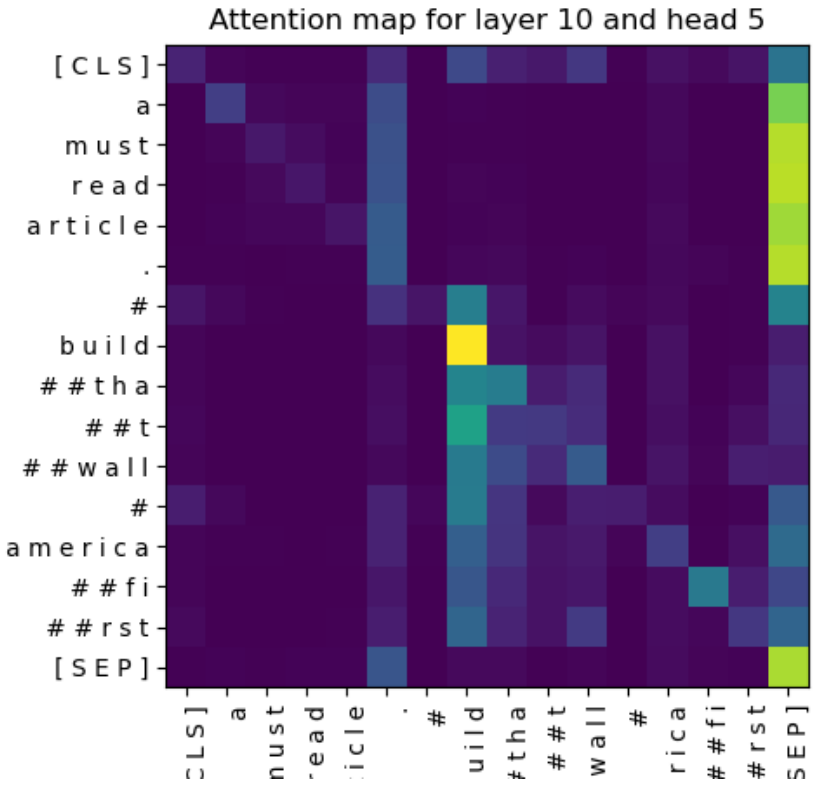

Figure 7.9

7.4.3 Effective attention

Even if attention maps have proved themselves to be a powerful instrument in debugging and detecting bias, according to some recent publications there is some room for improvements, in introducing the concept of effective attention maps. Effective attention maps are the maps obtained from attention weights, after that, a mathematical transformation has been applied, and they are considered an evolution of raw attention maps.

Effective attention is defined as the amount of attention that is effectively used by the following phases of the neural network or, mathematically speaking, as the difference between attention maps and the attention matrix that projects into a null space, e.g. the kernel of the matrix product between attention maps and the embedding matrix, the value matrix and the heads matrix.

Letting E be the embedding matrix, V be the value matrix and H be the heads matrix, we can define the variable T as the product of all these factors:

$$T = E \times V \times H \tag{7.1}$$

Using this equation, the effective attention can be defined as:

$$EA = A - \text{ Projection }_{\text{LN}(T)} A \tag{7.2}$$

Where LN indicates the null space of the matrix T, and what is obtained is thus the matrix A itself minus its projection over the null space of T. During the implementation, the null space projection has been obtained by multiplying the matrix A for the associated singular vector of T, which corresponds to the left null space basis.

For this reason, the implementation of a snippet able to visualize the effective attention maps required the clone of the *HuggingFace* project and the edit of the source of several files, since the matrices V and H are designed for internal use and are not an output of the default implementation.

Effective attention maps share most of the advantages disadvantages of raw attention maps. Indeed, in most the cases, the difference is not exist at all, since the attention present in the attention maps was totally used for the computations. In almost all experiments conducted, the resulting attention maps after the application of the mathematical transformation showed the same patterns as before the application of the transformation, but without the vertical line corresponding to special tokens, such as *[SEP]* or *[CLS]*. In a few cases, the effective attention differed from raw attention in a way that allowed to visualize new patterns, as showed in the figures 7.10 and 7.11, that illustrate the difference for the sentence "**To the bitch who said the United States is rape FREE. Ask the millions of women who haven't reported it due to how bad our society is.**", considering the last layer and the eleventh head:

In this specific case, is more evident from the effective attention maps that many tokens don't play any role in the determination of the class label, and the *[SEP]* token has less importance than what could have been emerged before the transformation.

In many other cases, however, these differences are not appreciable. An example can be found when considering the same sentence above, and the same head illustrated in the figure7.8, but using effective attention in place of raw attention:

In this specific case, only a few practical differences are ascertainable when compared with figure 7.8. The main difference is the drop of the vertical line over the *[SEP]* token, and the difference in the gradient color of the image is just a consequence of the change in the range of values that compose the image.

Another example is in the figure 7.13, where the focus on the word "migration", which is probably driving the decision of the classifier, is more visible in the effective attention map. In this case, the 10th layer and 5th head are shown.

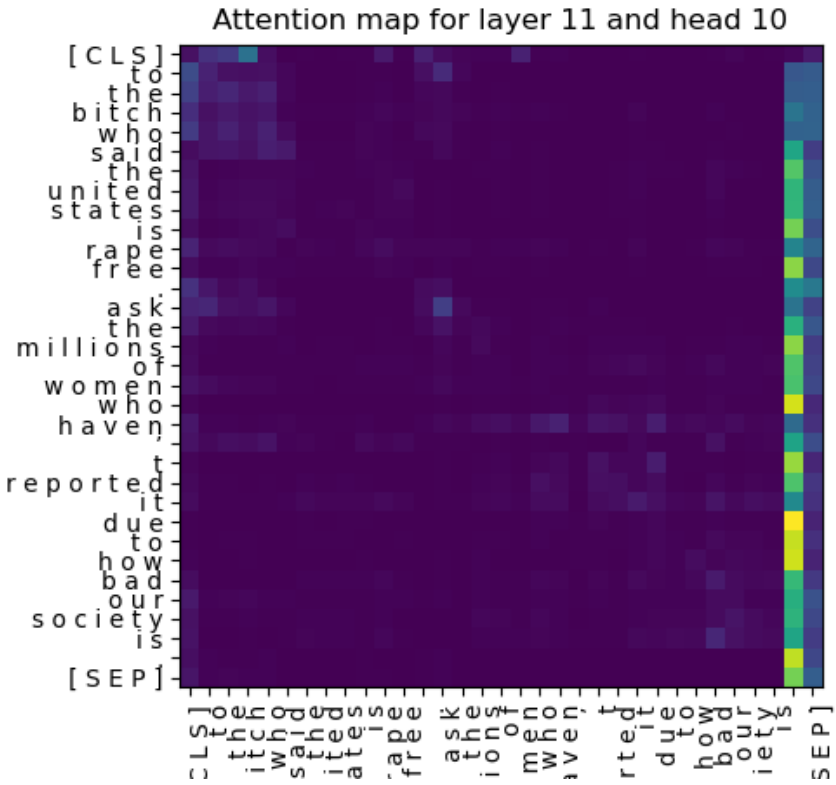

Figure 7.10: Raw attention

In conclusion, raw attention maps and effective attention maps share most of the advantages and of disadvantages. As raw attention maps, also effective attention maps proved to be a powerful instrument for debugging and detecting bias, and in some rare case, it provided a different point of view for an attention map, by highlighting the portion that was effectively used and making clear the portion that was not used for the determination of the class label.

Effective attention maps can be particularly useful when there is the need of investigating bias in all the tasks that expect the handling of sentences with a large number of tokens, such as next sentence prediction, question answering, and named entity recognition.

Indeed, as demonstrated by Brunner et al., the probability that effective attention diverges from raw attention is proportional to the number of tokens [30]. For this reason, since the experiments conducted concerned the classification of tweets, which were composed of a limited amount of tokens, the divergence between the two maps has been minimal.

In all similar cases, the use of raw attention maps should be preferable because

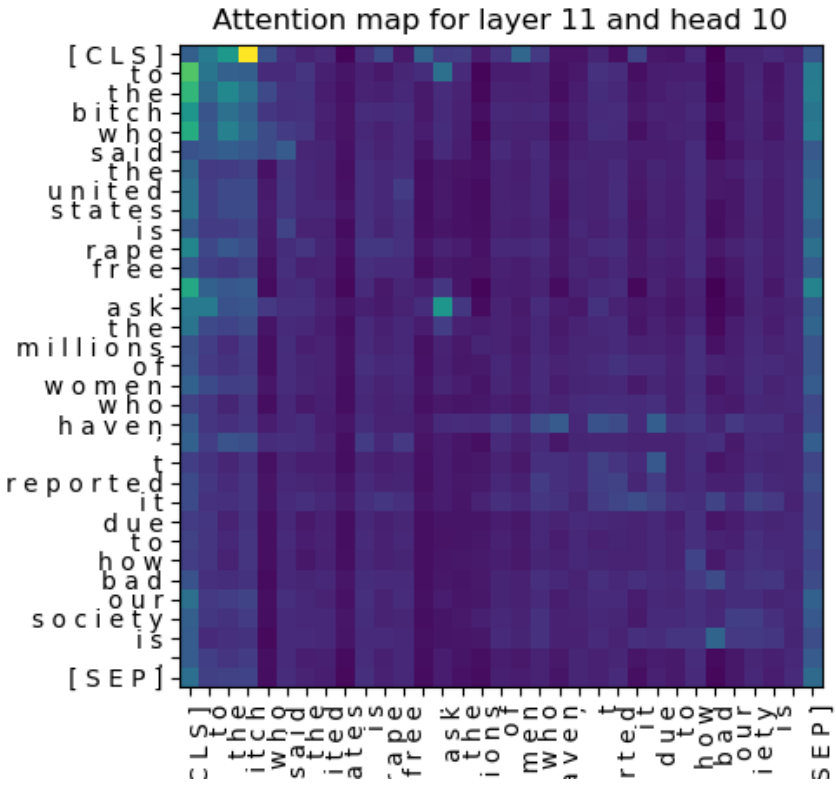

Figure 7.11: Effective attention

in most cases they provide the same information as effective attention maps without the need for further modification of the source code or the additional time required to calculate the mathematical transformation. This is because the raw attention maps can be computed in a period of time equal to the time required for inference. The effective attention maps, on the other hand, require an additional step, and given the computational cost of the SVD calculation, the time difference between the creation of the two maps is not negligible.

For this reason, the use of effective attention maps may not be worthwhile, except in a few exceptional cases.

7.4.4 SOC

SOC stands for Sampling and Occlusion and is a novel explanatory algorithm that estimates the post-hoc importance of features. Unlike other built-in methods such as attention maps, which rely on internal information to obtain an explanation from a neural network, SOC is model agnostic and context-independent. Context

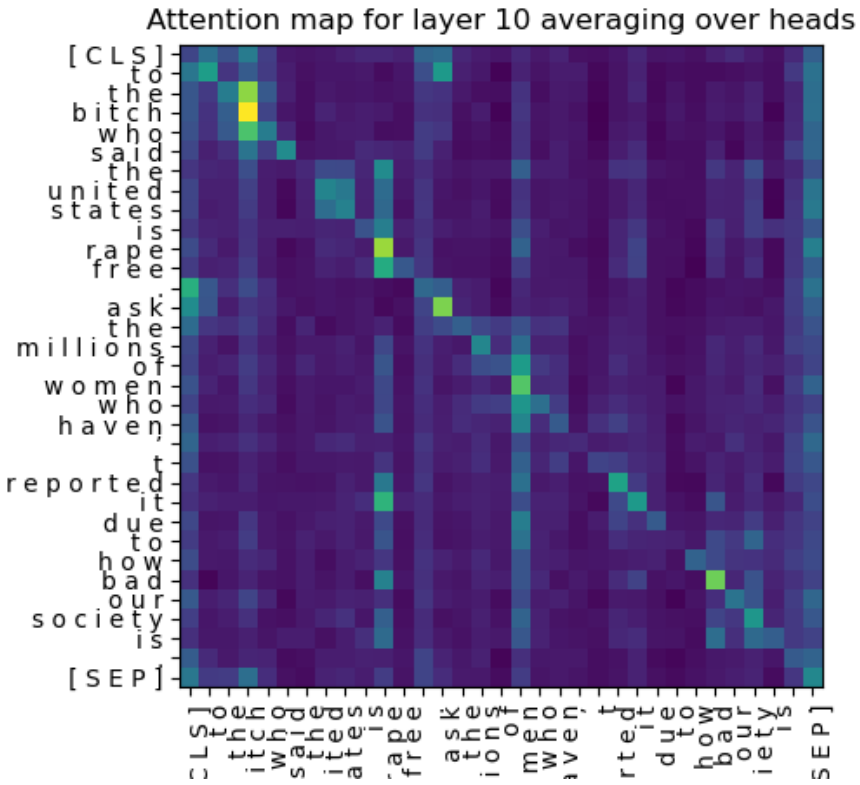

Figure 7.12

independence is a property of hierarchical explanation methods that classify each subsentence independently of the previous context given by the placement of the extracted tokens in the original sentence.

The philosophy behind the development of SOC is that an explanation can sometimes be a composition of conflicting facts that combine and compete to determine the classification of the input. For this reason, SOC proposes an explanation achieved by a hierarchical representation of the sentence obtained by sampling the original sentence to obtain different combinations of tokens, and by occluding the tokens that are not used in a particular level of the hierarchy. Hence the name "sampling and occlusion". On the other hand, all the explanation techniques presented so far in this thesis are based on assigning a score to each token in a sentence.

SOC, thanks to its hierarchical representation of the explanation, is able to appreciate different interactions between features without requiring the user to manually search through a large set of feature groups. Indeed, the SOC algorithm automatically selects the most meaningful interactions from a large number of

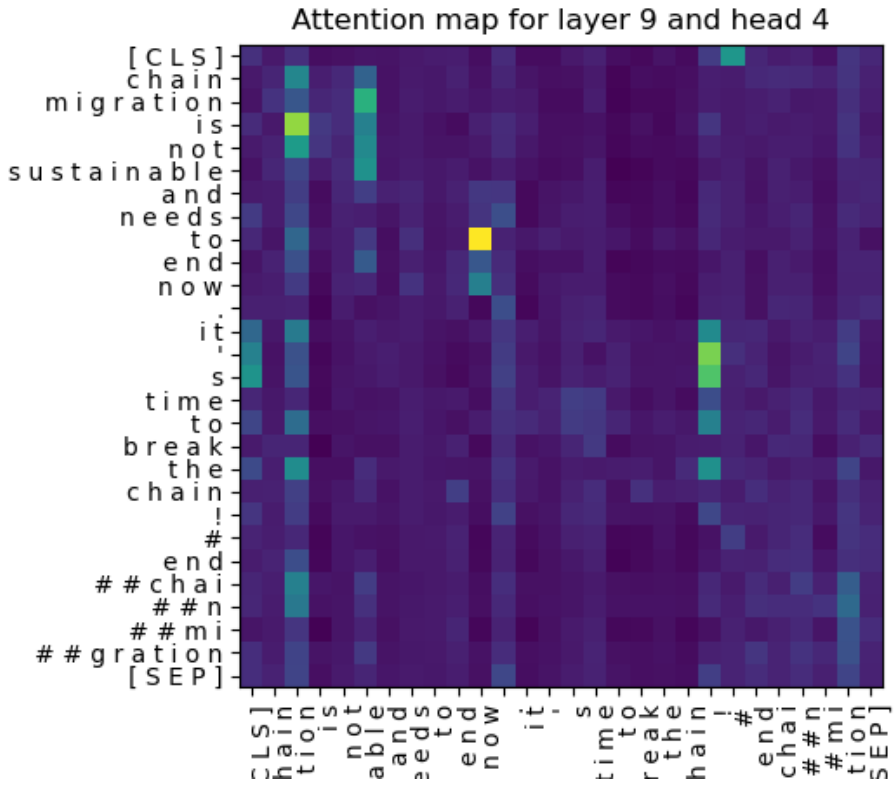

Figure 7.13

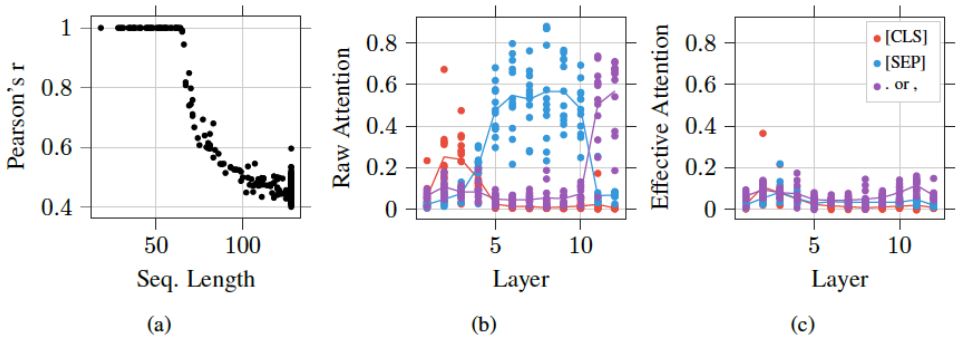

Figure 7.14: a) Pearson correlation index between effective attention and raw attention, over different token lengths. b) and c) respectively raw and effective attention, where each point represents the average attention of a given head to a token type. Figure taken from [30]

all possible combinations of feature groups and visualizes them graphically in a clear and meaningful multi-level table. Finally, the selected features are clustered to better appreciate a different score for the combination of feature groups than for the same feature groups taken individually. Many researchers from different fields agree that a hierarchical explanation is preferable for the end-user because it satisfies the characteristics of informativeness and limited essential detail [31]. For all these reasons, SOC is considered one of the best tools for the explanation when the classification is given to the final user, and it makes user-friendliness one of its main strengths. On the other hand, the production of classification for different hierarchical levels, and the search for the optimal subset of features to show at each level, comes at a cost: SOC may be excessively slow, also for producing the explanation limited set of inputs. Besides the time required for the computation of the output, also the time required for the adaptation and optimization of the SOC algorithm for customized tasks has to be taken into account. Indeed, SOC requires the fork and edit of an already implemented version of this algorithm, and for this thesis, the choice fell on the one implemented by Kennedy at al. [32].

The implementation of custom SOC settings can be divided into 3 different steps:

- **Training of a language model**, on the corpus, extracted from the sentences used for the classification. The language model is used during the step of occlusion to provide a background to substitute words
- **Training of the model**, which provides an inference for each input
- **Actual SOC algorithm**, which combines the two previous models in order to obtain a hierarchical representation

A further step is required to transform the metadata obtained in the last step into a graphical representation. As explained in chapter 5, SOC requires two different hyperparameters, namely the size of the context region N and the number of samples K, following the results obtained in similar experiments, both the parameters have been set to 5. After the implementation, the lead time required for all the steps amounted to 23 minutes, as executed on a *NVIDIA Tesla K80* offered by *Google Colab*.

The visual output proposed by SOC can be described in the following way: every feature group has a different background color, if the background color is gray it means that the tokens do not contribute to the determination of the class label. All the groups having a red background are pushing the classification of the whole sentence towards the positive class, and more intense is the red more relevant is the effect of the single group for the decision. Similarly, all the groups having a blue background are pushing the classification towards the negative class,

and also in this case a different intensity of the color is equal to a different effect of the sentence upon the decision of the class label. Sometimes, the explanation at different levels can be contradictory, and it can present a sub-sentence classified as positive in a level and, aggregating the sub-sentence with other tokens, the resulting composition can get assigned the negative class.

This is the case of the sentence **Nah yall stay in the kitchen and make bariis and baasto**, which is a false negative in our setting of the experiment, as showin in figure 7.15.

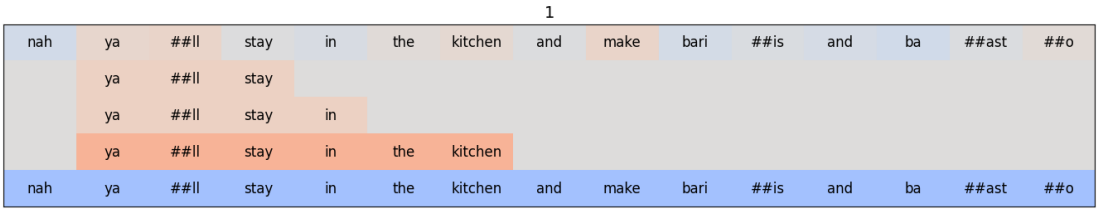

Figure 7.15

As it is clearly possible seeing analyzing the different levels of the output, the sub-sentence "**yall stay in the kitchen**" is classified as positive, and it then becomes negative in the last level. Differently than previously thought, the sentence "stay in the kitchen" is correctly classified as misogyny when taken alone, and the reason of the false negative is not that the dataset did not present any other example where women were invited to stay in the kitchen, but it is a result of the combination of the misogyny sub-sentence with the last part of the sentence, that balance the attribution towards the negative class, and makes the overall classification negative.

Another similar example can be found in the following false-positive sentence: "**@benshapiro Fuck you pussy**", as shown in figure 7.16.

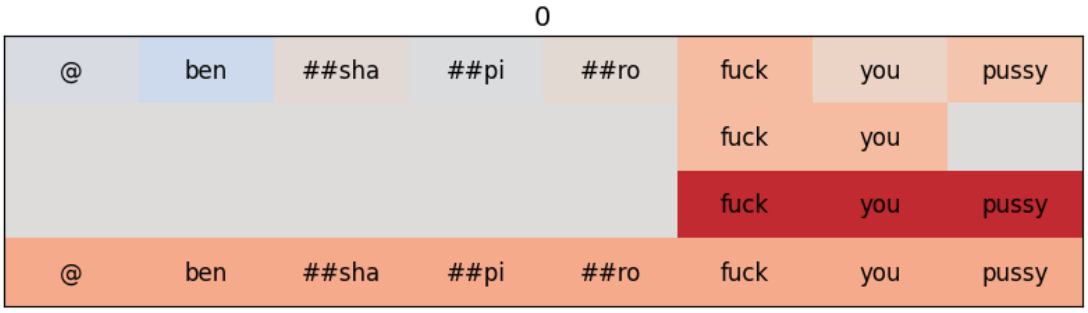

Figure 7.16

In this case, incredibly, SOC demonstrates how the neural network is able to understand that the name "Ben" in the tag "@benshapiro" is a masculine name,

and thus the background color of the "Ben" token is slightly tending towards blue. However, the impact of the features group "fuck you pussy" is too strong, and completely balances the negative class tendency introduced by the "Ben" token, ending up inverting the class label.

The analysis of the sentence "**@minimaslany What if she enjoys it? I think the days where women are forced to stay in the kitchen, not work, etc are over.**" is proposed, which is another false positive, and the result has been reported in the figures 7.17 and 7.18.

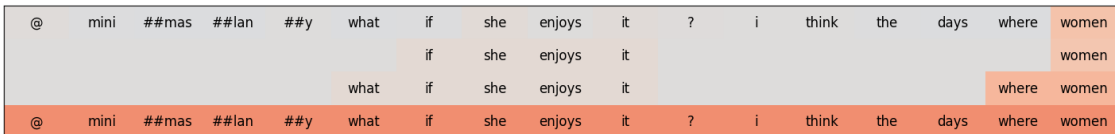

Figure 7.17

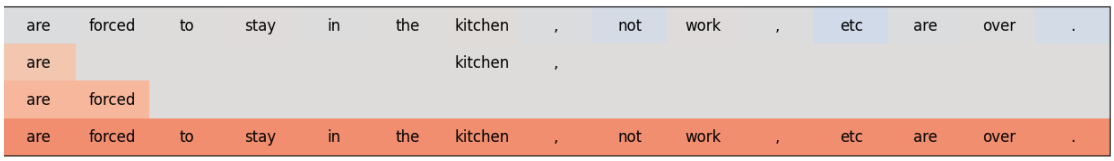

Figure 7.18: *Note: the sentence was split into two different images for readability reasons*

The explainer output is showing how the sub-sentence "what if she enjoys it" is classified as positive, as it is supposed to be, but the neural network is failing at understanding that the tokens are contained in a rhetorical question, and thus the significance has to be inverted. Similarly, "where women are forced" is considered a misogyny sub-sentence, and this is almost totally led by the single token "women". The same pattern has been found in other sentences containing the token "women". The tokens "not" and "etc" are considered to have a negative effect in the determination of the class level, but their impact is not of any significance at all, neither in the neural network nor in the real meaning of the sentence. In conclusion, given an importance score for each sub-sentence in a sentence, the most relevant contribution of SOC is in resolving the problem of finding a small meaningful set to show to users, through the many possible groups. In this analysis, SOC has been demonstrated as a powerful instrument in detecting bias, and its user-friendly output makes it ideal for the externalization of the checks of the neural network to not insiders employees. On the other hand, its long execution time does not make it ideal for the analysis during the development.

7.4.5 HTA

HTA is an acronym that stands for Hidden Token Attribution, and it is a quantification method based on gradient attribution. In the original paper [30], HTA was used for investigating how much each token does contribute to the following layer of the BERT architecture, and in which measure the information of tokens mix up in the last layers. However, in this thesis, HTA is used to obtain the explanation from a neural network, by calculating the contribution of each token towards the *[CLS]* token contained in the last layer of the architecture, which is ultimately used for the classification. The HTA formulation has been used to calculate the attribution over the last *[CLS]* with respect to each of the embeddings composing the input.

This choice is justified by the fact that, since the HTA shows how much of the input token is contained in a given hidden embedding, if the considered hidden embedding is the last *[CLS]* token, then the value is also a measure of how much the input is influencing the decision, thus an explanation of the algorithm. The HTA contribute for the of the hidden embedding e_j with respect to the input x_i, is given by the following formula:

$$c_{i,j}^{l} = \frac{\left\| \nabla_{i,j}^{l} \right\|_2}{\sum_{k=0}^{d_s} \left\| \nabla_{k,j}^{l} \right\|_2} \quad \text{with} \quad \nabla_{i,j}^{l} = \frac{\delta e_j^l}{\delta x_i} \tag{7.3}$$

Where l indicates the chosen layer.

Since the gradient is an output of the inference, the calculation is reduced to the multiplication between the gradient and the input, implemented through the retaining of the gradient during the inference, and the use of the torch utility *torch.autograd.grad*.

After applying the norm, the *softmax* function is executed over the values, and finally, the tokens *[CLS]* and *[SEP]* are removed to prevent these tokens to change the scale, since they can assume out of scale values that could distract the focus from the actual words of the sentence. The HTA values are addictive because of the *softmax* function that normalizes the values and enforces the property. However, in the proposed examples this property is not respected since the removal of the special tokens. The output of the HTA algorithm is a vector of values, one value for each token in the sentence. This vector has been organized in a table, and the background has been colored using a gradient. The more intense is the color and more the token is important with respect to the determination of the positive class.

The first example where the HTA algorithm has been applied is the sentence "**Back to the kitchen you stupid sandwich maker**", which is a false negative, and the result is shown in figure 7.19.

In this example, the tokens "stupid", "sandwich" and "maker" are of greater importance with respect to the other tokens. However, HTA is not able to explain

back	to	the	kitchen	you	stupid	sandwich	maker
0.05	0.07	0.05	0.08	0.07	0.12	0.11	0.14

Figure 7.19

why the sentence was wrongly classified as negative, since the importance of the tokens has been correctly assigned, focusing on the relevant part of the sentence. Probably, the deletion of the special tokens played a role in modifying the relative importance of tokens. In any case, more consistent behavior is expected by an explainer. There is a little change in the quality when considering false positives, as the sentence "**To the bitch who said the United States is rape FREE. Ask the millions of women who haven't reported it due to how bad our society is.**". The result is reported in the figure 7.20.

to	the	bitch	who	said	the	united	states	is	rape
0.03	0.06	0.16	0.04	0.05	0.02	0.02	0.02	0.02	0.06
free	.	ask	the	millions	of	women	who	haven	'
0.03	0.03	0.04	0.01	0.03	0.01	0.03	0.01	0.01	0.02
t	reported	it	due	to	how	bad	our	society	is
0.02	0.03	0.01	0.02	0.01	0.01	0.02	0.03	0.03	0.01
.									
0.03									

Figure 7.20

The tokens of greater importance are "bitch" and "rape", with the first having relative importance more than two times the latter. As opposed to the previous case, the explanation provided by HTA is reasonable. Indeed, "rape" is a word that is rarely seen in a non-toxic context, while the word "bitch" in this context is used to appeal to the person the user was speaking to, and the error committed by the neural network is understandable, since also for a human explainer this would be considered an edge case.

The explanation of the sentence "**@CraigRSawyer In ceuta Spain at the borderRefugees go home**", extracted from the dataset concerning the hate speech against immigrants, is shown in the figure 7.21. According to the hidden token attribution algorithm, most of the information driving the decision is derived from the "@" symbol, which is probably more represented in the posts where an argument is ongoing. Other tokens that got the highest importance scores are the "border" and "#gee", coming from the word "refugee". In the last example is analyzed the sentence "**Dont call them hysterical women.**", and the output is shown in the figure7.22.

Similar to the previous case, the HTA algorithm correctly individuated the token that, in all likelihood, deceived the neural network and that made the decision fall

@	craig	##rsa	##wy	##er	in	ce	##uta	spain
0.078798	0.048297	0.029021	0.03976	0.021668	0.024714	0.037488	0.050148	0.046713
at	the	border	##re	##fu	##gee	##s	go	home
0.033052	0.03183	0.076471	0.046705	0.062527	0.113317	0.038241	0.042542	0.05256

Figure 7.21

don	##t	call	them	hysterical	women	.
0.10	0.09	0.07	0.08	0.22	0.16	0.08

Figure 7.22

back to the positive class.

The focus is primarily upon the token "hysterical", which is a word that is rarely seen out of a misogyny context. On the other hand, the tokens composing the word "don't" did not receive an adequate importance score, and this could explain why the sentence was misclassified. In conclusion, since once obtained the gradient the calculation is simple, and the overall implementation of the algorithm is trivial, the ease and the speed in the use of HTA are two of its main strengths. In contrast, in some cases, HTA has demonstrated not being able of detecting the bias, and this reduces the utility of the tool. Particularly, as already seen in the first example, the HTA algorithm tends to show peaks of importance scores even when these tokens are not actually driving the decision, and this makes HTA a scarce tool when is about analyzing the false negative. Because of its poor reliability, the quality of the explainer is low.

7.4.6 KernelSHAP

The SHAP algorithm is an algorithm based on the game theory that sees the features as competitors in the determination of the class label. KernelSHAP is the base implementation of the SHAP algorithm, that makes large use of kernels, as described in the previous chapters.

The KernelSHAP implementation used for this thesis is the one created by Scott Lundberg and published on the PyPi repository under the name of *shap*. However, this version of SHAP was thought for the explanation of small tabular datasets, and the adaptation towards the NLP scenario has been challenging because of the re-definition and optimization of the problem.

First of all, the KernelSHAP implementation requires a pandas DataFrame with a fixed number of columns, that represent the features. This is incompatible with the structure of sentences, that have different amounts of words. For this reason, a maximum word length for the sentences has been decided and set to *64*, then each sentence has been split over the *space* character, and each word composing the

sentence has been put into a vector. To have all the sentences of the same length, a definite amount of empty word characters have been appended to all the vectors whose size was less than 64. In this way, each vector has exactly 64 words and the set of all the vectors can be seen as a table. The custom implementation of SHAP for BERT has been decided to work a word-level and not a token-level because the definition of features did not require in any way that those features had to be the same ones feed to the actual neural network. A further step has been introduced in the SHAP pipeline in order to translate words to tokens in the actual forwarding, as described later in this section.

The KernelSHAP implementation requires the definition of two hyperparameters, namely the size of samples composing the background and the regularization. The second hyperparameter has been to *aic*, while the first hyperparameter has been set to 100. The reason why a so small amount of samples has been decided for the background is explained by the high execution time and computational cost that characterizes SHAP. Indeed, also with 100 samples, the total amount of time required to explain ten sentences exceeded ten hours. The sentences list has been divided into two groups, the first one, named *train sentences* has been randomly selected from the test dataset, and they are then 100 sentences composing the background. While the second group, named *test sentences* was composed by the ten sentences, among false positives and false negatives, individuated previously and used all across this thesis.

The second obstacle has been identified with the type of data required from KernelSHAP. Indeed, KernelSHAP only requires float values into the DataFrame cells, and not the strings composing the words of the sentence. For this reason, the class *OrdinalEncoder* from *sklearn* has been used to encode each word into a float, and preserve the encoding rules in order to perform the inverse transformation when required. Both the groups have been joined before fitting the encoding, and then transformed singularly, in order to have a unique vocabulary across the experiment. The encoded words have been then inserted into a pandas DataFrame. Furthermore, since the SHAP implementation has no way to understand how to forward the input to the neural network since it's not meant for a neural network, a wrapper encapsulating the neural network has been created, which receives in input the combination of features generated by SHAP and returns the logits representing the classification.

The wrapper takes in input the model, the tokenizer and the encoder described previously. The model is used for generating the logits, the tokenizer is used for translating the words into tokens, and the encoder is used for applying the inverse transform, and transforming the encoded real values into their original string representation. For each combination of features generated by SHAP (in this experiment the amount exceeded 10 million), a special callback of the wrapper is called.

The first step is the application of the *inverse_transform* function of the *OrdinalEncoder* class, this reverts the vector of floats and obtains back the original vector of strings. Then the vector of strings is joined into a single sentence, using the *space* character. At this point, the sentence can be meaningful or meaningless, according to how much SHAP mutated the original feature list. Following, the tokenizer is used to obtain a BERT representation out from the sentence, and the list of tokens is then fed to the neural network, which outputs the logits. The logits are then returned to SHAP, and they are used for the internal calculation as described in the original paper. In the end, two NumPy matrices are returned from the *shap_values* function of *shap*, one containing the contribution to the negative class and the other one containing the contribution to the positive class. Since this is a binary problem with two mutually exclusive classes, the content of one matrix is the opposite of the other. Each matrix has a shape $10, 64$, where the number of rows indicates the test sentence and the number of columns indicates the token for that sentence. For this explanation, the interest is towards the words driving the decision to the positive class, so only the matrix referring to the positive class has been used.

In the first example, the sentence **Fuck yeah you pussy boy** is analyzed in the figure 7.23.

fuck	yeah	you	pussy	boy
-0.01181098	-0.11886397	-0.01541147	0.14337903	-0.01509997

Figure 7.23

In this example, is visible how the word "pussy" has the greatest SHAP value, and it is driving the decision towards the positive class. Since all the other words have a negative impact, including the word "fuck", thus "pussy" alone is determining the whole assignment of the class label.

In another example, the sentence **Why do y'all blame the woman for what she wears when it comes to rape? That shit is WRONG.** is analyzed, and the result is reported in the figure 7.24.

Why	do	y'all	blame	the	woman	for	what	she	wears
-0.0115063	-0.0246062	-0.0240694	0.0005342	-0.0164181	0.0258257	-0.0135379	-0.0103944	0.0274142	0.0162905
when	it	comes	to	rape?	That	shit	is	WRONG.	
-0.0116036	0.0096237	-0.0040296	-0.0126157	0.0120833	0.0006944	0.0020833	0.0006944	0.0006944	

Figure 7.24

In this example, the words "woman" and "rape?" are correctly individuated to be the ones driving the most decision towards the positive class. Note that,

differently than other examples in this thesis, this time the evaluation of the feature importance happens at word-level and not at token-level. For this reason, the importance of the word "rape?", retaining the exclamation mark, is evaluated in this case.

The last example is a false negative extracted from the dataset against the hate speech, and it is the sentence "**@whaas3 @judithineuropa Just got on twitter because of this farce today. Imagine this, I make a report on You. Calling you names and telling people how big liar and asshole you are without reason. Would you be angry? Of course You would be. Its a same with me. You can fuck off to help some rapefugee**", and the related output is illustrated in the figure 7.25. According to KernelSHAP, the focus is mainly on the tags present at the beginning of the sentence, altogether with the token "asshole", and not on the word "rapefugee". This could explain why the neural network failed to classify the sentence as hate speech against immigrants since "rapefugee" is slang composed of the words "rape" and "refugee" and not much of common use. Furthermore, KernelSHAP works at word-level, thus it is not able to recognize composing words, unlike other explainers that work at token-level.

@whaas3	@judithineurop	Just	got	on	twitter	because	of	this	farce
9.30E-02	1.56E-01	2.70E-02	-6.00E-03	-2.16E-03	-2.16E-03	-4.50E-03	4.43E-03	-1.16E-02	-2.35E-02
today.	Imagine	this,	I	make	a	report	on	You.	Calling
6.28E-03	-7.25E-03	-2.22E-03	-5.61E-03	4.12E-03	0.00E+00	-1.30E-02	0.00E+00	-2.20E-03	0.00E+00
you	names	and	telling	people	how	big	liar	and	asshole
0.00E+00	-4.99E-03	3.72E-03	-4.73E-03	-5.71E-03	6.63E-03	0.00E+00	-7.40E-03	-1.00E-02	1.07E-01
you	are	without	reason.	Would	you	be	angry?	Of	course
0.00E+00	3.85E-03	3.63E-03	0.00E+00	-2.34E-03	-3.66E-03	0.00E+00	4.42E-03	-6.56E-03	-1.09E-03
You	would	be.	Its	a	same	with	me.	You	can
-1.49E-03	-1.01E-03	1.97E-03	-2.27E-03	-4.31E-03	-1.02E-03	-1.58E-02	2.95E-02	-1.54E-02	-1.20E-02
fuck	off	to	help	some	rapefugee				
-2.43E-02	5.28E-03	-3.94E-02	1.99E-02	-2.36E-02	-6.18E-03				

Figure 7.25

KernelSHAP has been demonstrated to correctly individuate the most impacting words within a sentence, but the calculation arrived after a disproportionate use of time and resources. The result for the ten selected sentences only required more than ten hours on a T90 offered by *Google Colab*, and the use of the resources is not justified by a particularly enlightening explanation. Even if the correct words have been individuated, in a not a too dissimilar way to what happened with other explainers, the SHAP algorithm tends to assign a too high score to padding tokens, for a reason not entirely clear. Furthermore, all the changes introduced have no theoretical foundation and need to be proved to actually justify their use in production. For all these reasons, the result obtained from this experiment is considered unsatisfactory.

7.4.7 DeepSHAP

DeepSHAP is an extension of the SHAP algorithm which, basically, applies the Shapley values to the DeepLift algorithm.

The used DeepSHAP implementation is the one included in the *shap* package, which is internally based on the *deeplift* package. This SHAP extension makes use of the gradient information to provide a faster and better result, it is thus only indicated for neural networks, particularly for the ones based on *torch* that makes use of backpropagation. The DeepSHAP function, similarly to the KernelSHAP function, has required the implementation of a wrapper around the neural network, to correctly translate the features generated and sent by DeepSHAP into something to feed the neural network. Among the other things, the DeepSHAP needs two elements to compute the gradient and thus the DeepLift values, and they are the logits emitted from the neural network and the embedding fed as input. This required two little changes from what was used so far. First of all, the neural network source has been modified to output the embeddings, thus the embeddings (and not the tokens) are fed to the neural network. Because of the nature of BERT, and the fact that DeepSHAP was not designed for working with NLP architectures, a little change has been applied to provide compatibility. The edit was applied to the DeepSHAP source code and it concerns the skip of two types of layers in the accounting of the contributions of the DeepLift values, and they are the layer types *linear_1d* and *nonlinear_1d*.

DeepSHAP requires only one hyperparameter, it is the background samples size and it plays the same role as in KernelSHAP. In the experiments performed, the value has been set to 40 for performance reasons.

In the figure 7.26, the analyzed sentence is the false positive "**@benshapiro Fuck you pussy**".

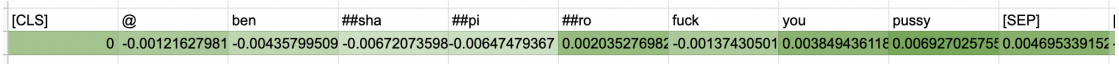

Figure 7.26

In this sentence, DeepSHAP correctly individuates the word "pussy" as of greater importance for the explanation, and correctly assigns a negative score to the masculine name "Ben". However, the word "fuck" is weirdly considered as negatively impacting the negative class.

Another example is presented in the figure 7.27, where the sentence analyzed is the false positive "**Fuck yeah you pussy boy**".

Similarly, the word "pussy" is considered an important token for the determination of the positive class. However, differently from the previous example, the word "fuck" is playing an even more decisive role, being considered of greater importance

[CLS]	fuck	yeah	you	pussy	boy	[SEP]
0.0000000	0.0144948	0.0007756	0.0016419	0.0106113	0.0013996	0.0051093

Figure 7.27

than "pussy".

To sum up, DeepSHAP shares most of the advantages and disadvantages of KernelSHAP. The main difference is the execution time, which amounted to 8 minutes on a T80 GPU and at 32 minutes on an average laptop CPU. DeepSHAP offers an explanation similar to KernelSHAP, but since it required less impacting modification on the core of the algorithm to be adapted on BERT, the theoretical foundation that bases the working are stronger in this context. Even if the quality is low in both of the explainers, if a comparison between the two has to be performed, DeepSHAP may be preferred over KernelSHAP.

7.4.8 MiCE

MiCE stands for Minimal Contrastive Editing and it is a novel algorithm for producing contrastive explanations of model predictions. MiCE is meant for providing an explanation of NLP neural networks, and its initial implementation was based on BERT.

One of the biggest innovations brought by MiCE in the form of the provided explanation. Indeed, MiCE does not provide an importance score for each token of the sentence, as other explainers. Instead, MiCE provides an explanation by producing a list of counterfactual sentences that flip the decision. If a sentence is classified as positive by the neural network, thus, MiCE tries editing each word of the sentence, then couples and triples, until the edited sentence is classified as negative. There are some constraints that edits must respect: they have to be minimal, thus not longer than required and as similar as possible to the original sentence, and they have to be fluent and natural. If the produced sentence makes no sense at all it is then discarded, differently than what happens with KernelSHAP.

The implementation of MiCE used for this thesis is based on a fork of the *allenai* work, in the GitHub repository under the name of "mice". The changes applied to the original code include the addition of the used dataset, and a couple of optimizations required to reduce the computational cost of the explanation, such as the decreasing of the batch size and the limit of the number of training samples. These changes reduced the quality of the explainer but allowed the execution in reasonable times. Indeed, among all the trained explainers, MiCE has by far the highest computational cost and the highest lead time. The first stage of MiCE, which trained the editor, required more than 10 hours, while the second stage exceeded 12 hours. Thus, the total amount of time for the explanation of the whole

test dataset amounted to more than 22 hours.

After the second stage, a further step is required to transform the generated metadata into a human-readable piece of output. And it can be either visualization of the generated contrastive edits in a notebook, or the generation of a comma-separated values file that contains a list of different contrastive explanations for each sentence.

The CSV file was mainly used for this thesis, and it presents a list of all the sentences included in the test dataset, and for each sentence, a variable number of contrastive edits applied to flip the decision. Each contrastive edit is then sorted from the most minimal to the least minimal.

The word or the set of words that caused the flip is not immediately indicated, and they have to be deducted by observing the difference between the edited sentence and the original sentence. In the example visible at the table 7.1, the list of the first ten contrastive edits applied to the sentence "**Fuck yeah you pussy boy**" are shown.

Fuck yeah you idiot boy
Fuck yeah you dude boy
Fuck yeah you drunk boy
Fuck yeah you stupid boy
Fuck yeah you a boy
Fuck yeah you fat boy
Fuck yeah you jady boy
Fuck yeah you bloody boy
Fuck yeah you creepy boy
Fuck yeah you cocky boy
Fuck yeah you sneaky boy

Table 7.1

The list of sentences proposed by MiCE, as visible from the reported table, is contrastive since they present the opposite meaning of the original sentence. Even if MiCE does not propose an importance score for each token, from the list presented above is possible seeing that only the word "pussy" is changed across all the sentences, thus one could conclude that it is the feature of greater importance for the determination of the positive class.

All the proposed contrastive sentences are fluent and minimal, and they are sentences that totally would make sense in a not misogyny context. Another example is reported in the table 7.2, where the contrastive sentences related to the false positive "**Dont call them hysterical women.**" is presented.

Here again, ten contrastive sentences are reported, and they are sorted from

Dont call them idiotic women.
Dont call them stupid women.
Dont call them real women.
Dont call them fuck women.
Dont call them fucking women.
Dont call them ugly women.
Dont call them hot women.
Dont call them fat women.
Dont call them racist women.
Dont call them bad women.
Dont call them lyrical women.

Table 7.2

the most minimal to the least minimal. In this case, as in the one reported in the table 7.1, the same word for each proposed sentence has changed, and it is the one related to "hysterical". This demonstrates that, in order to obtain the classification towards the negative class, the only word that needs to be changed is "hysterical", and not "woman" as someone could suppose.

In the end, a false negative is analyzed. The contrastive sentences for the example "**Nah yall stay in the kitchen and make bariis and baasto**" is analysed in the table 7.3

Nah girlsall stay in the kitchen and make bariis and baasto
Nah women should stay in the Punjab and make bariis and baasto
Nah girls stay in the UAE and make bariis and baasto
Nah women stay in the mosques and make bariis and baasto
Nah women should get in the kitchen and make bariis and sabhaisasto
Nah women stay in the Muslim communities and make bariis and baasto
Nah womenall stay in the same room and make bariis and baasto
Nah women in the street and make bariis and wadeasto
Nah ladies in the streets and make bariis and bardisasto
Nah ladies in the desert and make bariis and baridasasto
Nah fuck them in the desert and make bariis and guptasasto

Table 7.3

With the only exception of the last contrastive sentence, all the first 9 proposed examples replaced the token "y" of the couple "y, #all" with either "women" or

"ladies". This demonstrates how the absence of a reference to the female sex was the main problem in the misclassification of the example. The sentence "stay in the kitchen", if asked to a human, is obviously addressed to women and would be without a doubt considered a misogyny statement. However, for a neural network, this could not appear as obvious, since there is not a reference to women and the invitation to stay in a neutral place like a kitchen may not seem as toxic as thought. Considering this, the mistake committed by BERT is overall understandable. The word "kitchen" has been sometimes changed across the proposed sentences and replaced with "UAE", "desert", and "street".

Analyzing both the training dataset and the test dataset, indeed, one could easily find that these are the place where a woman gets cursed to be raped the most in a toxic context, for this reason when MiCE looked for a physical place to substitute the word "kitchen" with, these ones get selected. This makes clear how "kitchen" is not a strong enough word for driving the classification towards the positive class. In addition to that, in most of the sentences proposed by MiCE, the word kitchen is replaced with "Punjab", "UAE", or "Muslim communities". This important fact demonstrated that the neural network assumed a bias towards the people living in these lands or communities. The assumed bias can be explained by the fact that these groups of people are the most mentioned when users use misogyny as an offensive weapon. These findings shed light on the exploration of bias in hate speech detection and demonstrate how a contrastive sentence can detect a bias out of the context that one was analyzing at that moment.

Finally, MiCE demonstrates to be a powerful tool for bias detection and correctly provided interesting insights in all the analyzed cases. Proposing the output as a list of contrastive sentences could be a mighty aspect for the analysis conducted by final users that are not insiders in the machine learning field. Its high time and resource requirements could be a disincentive in its use, but the originality and the quality of the result could justify the decision. The overall explanations provided by MiCE are thus considered satisfactory.

7.4.9 Comparison among explainers

In this section is proposed a comparison among the explainers using the same sentence to better highlight the differences among explainers and ascertain how different explainers treat in a more or less similar way the same token of that sentence.

The selected sentence for the comparison is **"@minimaslany What if she enjoys it? I think the days where women are forced to stay in the kitchen, not work, etc are over."**, since it offers different elements to investigate within the same sentence, such as rhetorical questions and a meaning flipping present in the last two tokens.

In the figures 7.28 and 7.29 are visible respectively the attention maps and the effective attention maps related to the selected sentence. In both cases, the weights composing the images are extracted from the 11th layer of the BERT architecture, and the average across the heads has been executed.

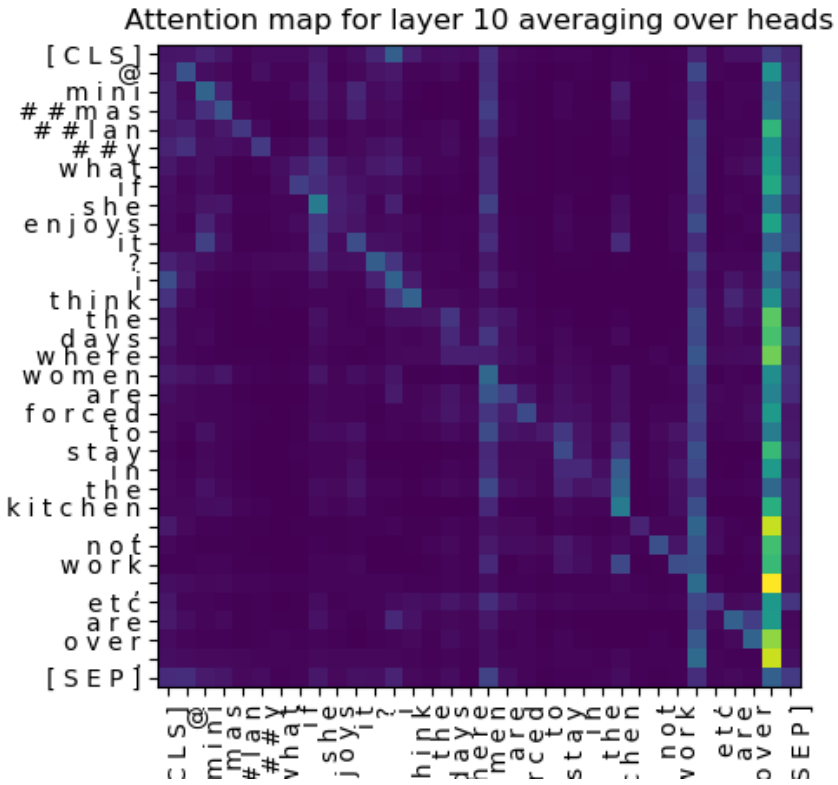

Figure 7.28: Attention map of the 11th layer, averaging across heads

From the images, is easily visible how most of the attention is focused on the token "women", while a smaller amount of attention is dedicated to the token "are over", which would be important for the attribution of the negative class. The same information is expressed by both the attention maps since any new pattern emerges from the effective attention map, if not a higher contrast that makes more evident the lines, caused by the a smaller amount of attention on the *[SEP]* token. Similarly, also when obtaining the explanation from SOC is visible a focus on the same tokens, as it's visible from the figure 7.30. However, in this case, even if the focus is evident on the single token "women", it is still greater if considering the group of tokens "where women are forced", and a relevant focus is also present for the group of tokens "what if she enjoys it".

A similar result was provided by HTA, visible in figure 7.31, where however

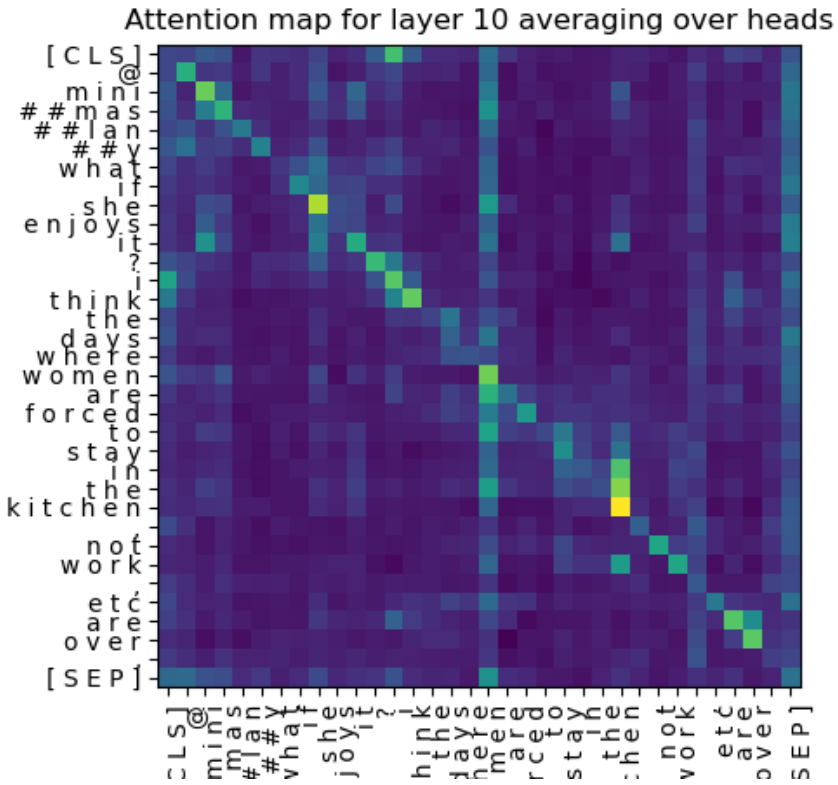

Figure 7.29: Effective attention map of the 11th layer, averaging across heads

Figure 7.30: SOC explanation

a high focus has been revealed also on the "@" token. This can be explained because, sometimes the neural network relies substantially on the tag present at the beginning of the sentence to determine whether the sex of the person with whom someone is speaking is female or male.

In the HTA explanation, a relevant importance has been attributed to the tokens "women", consistently with other explainers. Also the tokens "kitchen", "forced", and "enjoy" are relevant for the explanation according to HTA, and the token "etc.", for some not clear reason is playing a role.

In the DeepSHAP explanation, in figure 7.32 is clearly visible as the explainer struggled to provide meaningful insight into the neural network. In this case, according to DeepSHAP, the comma character is playing the most decisive role

@	mini	##mas	##lan	##y	what	if	she	enjoys	it
0.130164	0.051888	0.039956	0.025139	0.015133	0.01728	0.016783	0.0188	0.042103	0.013019
?	i	think	the	days	where	women	are	forced	to
0.023399	0.024721	0.022446	0.015878	0.029539	0.025191	0.083496	0.024106	0.031761	0.00951
stay	in	the	kitchen	comma	not	work	comma	etc	are
0.021199	0.01236	0.011475	0.048346	0.013827	0.016499	0.022964	0.014449	0.036699	0.015129
over	.								
0.013912	0.023498								

Figure 7.31: HTA explanation

in the attribution of the positive class. Obviously, this token is not playing any substantial role and thus the explainer cannot be considered reliable.

It's important to notice how also the padding tokens got assigned a relatively high Shapley value, while the expected amount for those tokens was 0. In this graphical representation of the DeepSHAP output, as well as in the KernelSHAP output, a yellow background has been assigned to tokens with a negative score to make easier the observation of the general trending. On the other hand, in KernelSHAP explanation (fig. 7.33), the clear pattern already seen in the other explainers can be observed. Indeed, the token "women" played the most decisive role according to this explainer, but a relevant role for the determination of the positive class is also determined by the "she" and "kitchen" tokens. In this case, the empty cells that filled the word count up to 64, got a Shapley value close to zero. For this reason, in this case, KernelSHAP is considered more reliable than DeepSHAP.

In table 7.4, the first ten contrastive sentences related to the selected sentence and produced by MiCE are shown. According to MiCE, the word "kitchen" is driving the decision to the positive class, and thus it must be replaced to obtain the classification of the sentence as not misogyny. It's important to notice how another set of tokens that is consistently changed across the contrastive sentences is the one related to the tag. Indeed, the original sentence reports the tag @minimaslany", which is not kept in any of the counterfactual explanations.

Analyzing the lower rows of the table, which concerns the less minimal contrastive explanations, is clear how also the token "enjoys" needs to be changed to reach the negative classification, and it is replaced often with "made" and often with "had", or other neutral verbs. In the end, the lead time to obtain an explanation, starting from the neural network already fine-tuned, is reported in table 7.5.

In this table, one can observe how attention maps, effective attention maps, and hidden token attribution scores have a lead time comparable with inference from the neural networks, for this reason, they can be used on the fly during the development, for testing the good nature of the dataset or of the pipeline. Instead, DeepSHAP and SOC, even if not immediate, can produce interesting results in a reasonable amount of time. Indeed just a few minutes are required for the explanation, and if that one is reliable and useful, can predictably be worth the

time consumed. Last, MiCE and KernelSHAP required hours for the execution, and this can be considered a problem whether the explanation needs to be repeated often or obtained in short times, or when there are not the resources required for the execution of the explanation.

With the only exceptions of DeepSHAP and MiCE, all the neural networks agreed that *Women* is the most important token for the determination of the positive class. However, the two exceptions are not in equal measure different. While DeepSHAP was not able to provide a meaningful explanation how the inner working of the neural network, MiCE aimed to produce a different type of explanation, which is counterfactual and not importance-based, and this can justify its different behavior. Even if MiCE provided a different explanation, does not mean that it was useless or did not provide important insight. Instead, MiCE can be considered complementary to importance-based explanations and used to observe aspects of the neural network, besides unintended biases, that cannot be observed in other explainers. For this reason, MiCE can be considered an important tool and must be used whenever possible. Effective attention maps did not provide any further useful piece of information besides the one already provided by raw attention maps. Since there is not a library that easily implements this transformation, the obtained explanation does not worth the used time for the implementation. Raw attention maps provided some useful insights, even if a single layer is not enough to express all the potential information contained in attention maps, not even when an average is performed across the heads, thus only twelve images need to be analyzed for an explanation. Raw attention maps are useful for detecting the biases and for exploring the inner working of the neural network, but their analysis should be left to developers and eventually proposed to not insiders as a report. For this reason, it's not clear if the attention maps can be defined as an explanation, as reported in the bibliography, or it is just a useful tool for investigation. In any way, their use is encouraged and justified, also considering their short lead time.

KernelSHAP has been demonstrated reliable in the experiments performed, even if the scarce theoretical foundations, caused by the changes applied to the algorithm, do not guarantee a good result in other tasks or other datasets. However, its use can be justified in case different explanations of different neural networks must be compared, and thus Shapley values are used as a common measure among the neural networks. On the other hand, HTA has been demonstrated to be reliable in all the analyzed cases and did not cast a shadow over its reliability under any circumstances. The HTA method is based on gradient attribution, which was not considered in any other explainers with the only exception of DeepSHAP, which however obtained lower performances.

Considering that HTA can be executed on the fly since the gradient information is an output of the neural network, and its implementation is also easy and immediate,

its use is encouraged and advised in almost all cases. In the end, SOC provided useful insight on the combination of tokens, which was missing in all the other explainers. The importance attributed to a single token is not enough for the detection of biases in many cases, because the importance of a part of the sentence is not given by two single tokens considered separately, but if and only if those tokens are combined. A useful investigation of neural network biases cannot be exempted by the use of SOC or similar hierarchical explainers.

[CLS]	@	mini	##mas	##lan	##y	what	if	she	enjoys
0	-0.00121627981	0.000653494770	-0.00507134158	-0.00255937723	0.003071293327	-0.00162738055	0.000263478383	0.003749529919	-0.00427619853
it	?	i	think	the	days	where	women	are	forced
-0.00822652372	-0.00043360044	0.003197336214	0.001166151191	-0.00146962137	-6.74E-05	0.003868815644	-0.00354153609	-0.00099539148	-0.00174316681
to	stay	in	the	kitchen	,	not	work	,	etc
-0.00303159961	0.001831485809	0.001073184939	0.001030810709	0.001950683273	0.008069557959	-0.00411075890	-0.00067588391	0.001906717886	-0.00019051898
are	over	.	[SEP]	[PAD]	[PAD]	[PAD]			
-0.00063410427	-0.00356636593	-0.00060144684	0.001347217735	-0.00000248065	-0.00052873580	0			

Figure 7.32: DeepSHAP explanation

@minimaslany	what	if	she	enjoys	it?	I	think	the	days
-0.015781949	0.0294738528	-0.0154056424	0.107317256	-0.0243021626	0.00527567076	-0.0394265911	0.0198692957	-0.0236313618	-0.0120438737
where	women	are	forced	to	stay	in	the	kitchen,	not
-0.00577975134	0.384051518	0.0427929726	-0.00789707597	0	0.0118387288	0.0081078123	0.00959477353	0.0678602331	-0.0428517488
work,	etc	are	over.	<empty>	<empty>	<empty>	<empty>	<empty>	<empty>
0.0147717919	-0.03526234	-0.0166404476	0.0160862716	-0.00690812125	0	0	-0.005678609	-0.00592172709	0

Figure 7.33: KernelSHAP explanation

@sharlenecrowson What if she enjoys it? I think the days where women are forced to stay in the streets, not work, etc are over.
@RenieSaunders What if she enjoys it? I think the days where women are forced to stay in the home, not work, etc are over.
@jakealannolly What if she made it? I think the days where women are forced to stay in the streets, not work, etc are over.
@BiraKaiser What if she made it? I think the days where women are forced to stay in the homes, not work, etc are over.
@realDonaldTrump What if she had it? I think the days where women are forced to stay in the cold, not work, etc are over.
@smaela_cannon What if she gets it? I think the days where women are forced to stay in the country, not work, etc are over.
@AlfredHenry What if she did it? I think the days where women are forced to stay in the house, not work, etc are over.
@ClaintyRomette What if she likes it? I think the days where women are forced to stay in the home, not work, etc are over.
@senjawmoon What if she harmed it? I think the days where women are expected to stay in the home, not work, etc are over.
@naeieditho: What if she said it? I think the days where women are forced to stay in the country, not work, etc are over.
@themrsamyassangedd What if she reacted like it? I think the days where women are forced to stay in the house, not work, etc are over.

Table 7.4: MiCE explanation

Att. maps	Eff. Att. maps	SOC	HTA	KernelSHAP	DeepSHAP	MiCE
0.046	0.048	1412.11	0.046	13 202.92	486.61	79 200.00

Table 7.5: Lead time to obtain execution. Time expressed in seconds

7.5 Analysis of vertical patterns

An analysis has been conducted to check if there's a correlation between vertical patterns and false-positive rates.

The analysis is justified by the fact that, often, a vertical pattern in an attention map is caused by a bias of the social network, that after the training became too sensitive to that specific token, and this can bring to a false positive. Thus, an experiment has been set up to prove or discharge this hypothesis.

In order to prove that a correlation subsists, an algorithm has been developed for the detection of the vertical patterns, thus for each attention map obtained from the inference of all the sentences of the test dataset, the vertical patterns have been counted. If that image belonged to a false positive, then the counter related to the false positive has been increased. Similarly, if that image belonged to a false negative, a true positive, or a true negative, the related counter has been increased.

The definition of vertical pattern was not previously expressed in a clear and objective way, not even in the paper that proposed the concept, where the early classification of whether a specific pattern was vertical or not was left to manual annotation.

For the purpose of this thesis, the concept of vertical pattern has been defined in the following way: a pattern is vertical with respect to a column if, for that column, the ratio between the values of the column above a certain threshold and the length of the column is above another threshold. The same concept, using α and β to indicate the two thresholds, can be defined mathematically in the following way:

$$\frac{\sum_{i=0}^{N} 1(v_i > \alpha)}{N} > \beta \tag{7.4}$$

In the above equation, N is the height of the column, $1(...)$ is a function that evaluates the condition and its value is 1 if it is true and 0 otherwise, v_i are the values contained in a column, α and β are parameters of the definition. The value of α has been obtained increasing the contrast of the attention map as much as possible, using the *torch* preprocessing features. Instead, the value of β has been voluntarily left unspecified and the change in the false positive count varying this parameter has been observed and analysed, in order to find the value that maximize the count for a given β.

If the count of the vertical pattern, as defined above, is greater among the false positives rather than other indicators, it means that a vertical pattern is more common when observing a false positive with respect to a false negative, a true positive, or a true negative.

To make the measures comparable, the count of vertical pattern related to false-positive has been divided for the false positive rate, and similarly, all the other counts have been divided for their respective rate. In this way, the measure

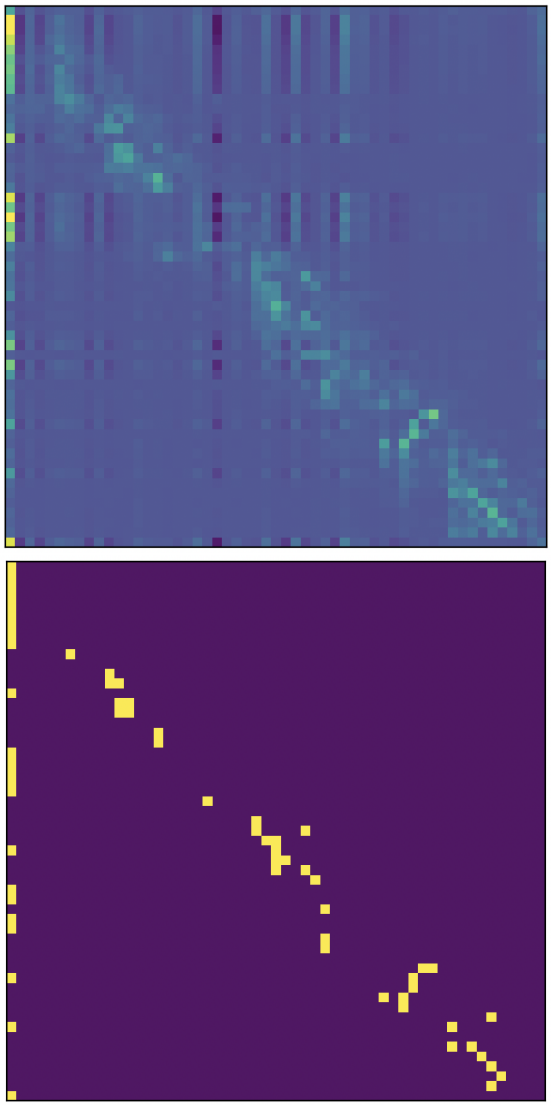

Figure 7.34: The same attention map before and after increasing the contrast

has been normalized and does not depend on the number of instances classified correctly or the error committed by the neural network.

The effective attention maps of each head, for each sentence of the whole dataset has been utilized. The count ratio has then been plotted using *matplotlib*, varying the parameter β. The result is visible in the figure 7.35. To make the values more comparable, a *log* transform has been applied to all the values before plotting.

As it is visible in the figure, independently by the β value, thus by the definition of vertical pattern, there is a difference in the count of false-positive and other indicators. The false positives and the true negatives appear to be strictly related

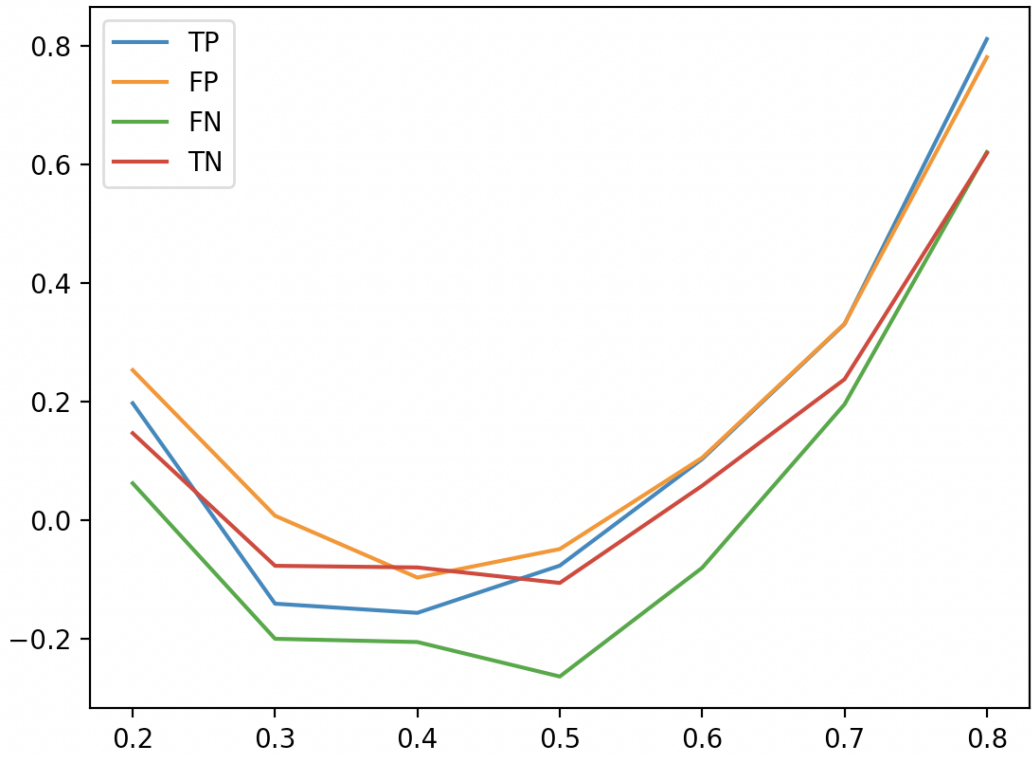

Figure 7.35: Plot of the normalized vertical pattern counts varying the parameter β

to vertical pattern, while the false negatives and the true negatives appear less related. Since in this thesis the bias is investigated, a focus on the only error has been performed, and in figure 7.36 only the count of the false negatives and the false positives is reported.

The fact that also the true positive are correlated to vertical pattern should not be surprising, since when the neural network assumes a bias towards a certain token chances are that in most of the cases the bias will lead to a correct prediction.

In the end, in the figure 7.37, the difference between the two curves has been plotted, and it is evident how, independently by the parameter β, there is a difference of 20% between the count of vertical patterns in false positives and false negatives. The difference is relevant and can lead to the conclusion that a correlation exists and it has been proved, even if with the limits of this research that has been executed on a single dataset. Even if vertical patterns are more common in false-positive samples, hardly this finding can be used to detect the bias, since the difference in the count ratio is not large enough to draw conclusions

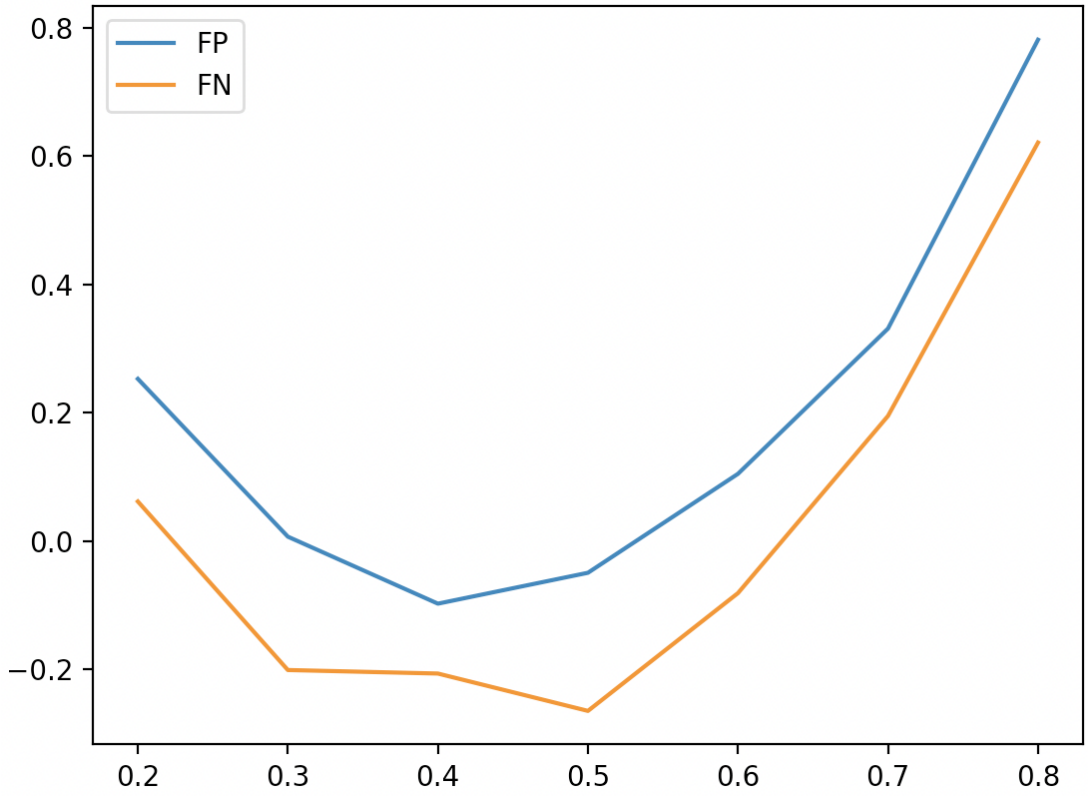

Figure 7.36: Plot of the normalized vertical pattern counts, focusing on false positives and false negatives, varying the parameter β

about bias beyond doubt.

A more extended research aimed to prove this hypothesis in other datasets or in other tasks is encouraged. The conditions leading to this result can be the subject of discussion and research.

7.6 Correlation among explainers

In this section, an exploratory analysis of the correlation among explainers is proposed. During the experiments performed for this thesis, many different explainers have been analyzed and used. Each explainer had a different internal working, and few similarities were found between each other.

Often, the explainers were based on similar concepts, as HTA and DeepSHAP, which both relied on the gradient attribution mechanism. In other cases, the proposed explainers were based on completely new concepts with few relationships

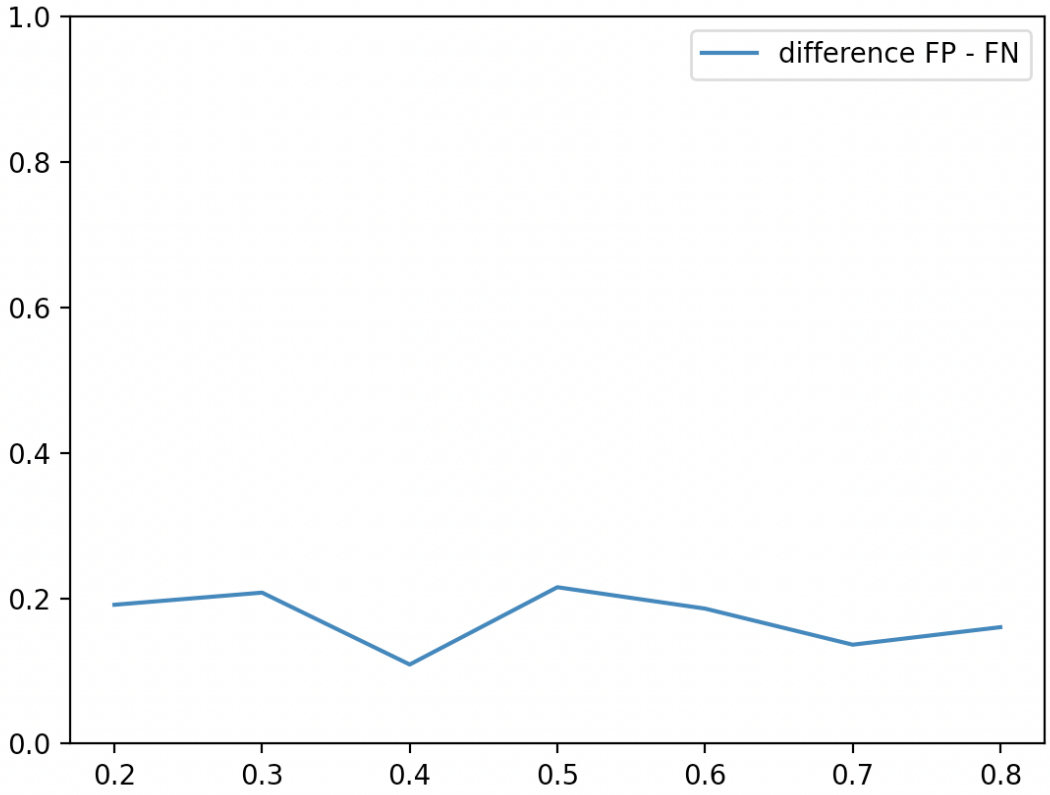

Figure 7.37: Plot of the difference of normalized vertical pattern counts related to false positive and false negative, varying the parameter β

with the other ones. For this reason, it's not clear if the explainers are expressing the same explanation, if two or more of the proposed algorithms just show two aspects of the same result, or if in some circumstances they can be considered equivalent.

If that would be the case, one of the algorithms could be considered redundant, and the use of both of them may be avoided since they would bring in a too similar result. On the other hand, if two explainers are loosely correlated, they can be used together, in a complementary way, because they are expressing two different types of correlation. This analysis has been conducted on the ten sentences selected in chapter 7.1, and it made large use of spreadsheets for annotating explanations and calculating the correlation. The annotation of the explanations has been performed manually, by observing the output of the different algorithms. The only two exceptions concern the attention maps and the effective attention maps. Because of the high number of images to analyze, an algorithm has been developed for printing, for each token in a sentence, what is the amount of vertical patterns

related. The correlation has been calculated filling a matrix, wherein the horizontal ax appears all the tokens composing each of the ten sentences, while in the vertical ax appear the 7 considered explainers. The matrix is filled with 0 if the explainer does not consider the token important, with 1 if the explainer does consider the token important, and with -0.25 if the explainer does consider the token as related to the negative class, in the case the explainer includes this eventuality.

The reason that led to assign -0.25 and not -1 for the negative explanation is due to the fact that, in that case, two explanations are very similar but provided by two explainers, one of which includes negative scores and the other does not, would result as completely different and uncorrelated. On the other, would be wrong to assert that the two explainers are strictly correlated. Assigning -0.25 in place of -1, instead, the two explainers would result in just slightly correlated, and this is the desired behavior. For simplifying the filling of the matrix, since some explainers as MiCE did not report a score, only the values 1, 0, and 0.25 were used, with no intermediate values.

In the case of MiCE, the words that have changed in the contrastive explanation got assigned 1, while the other ones got assigned 0. In the case of SOC, all the words having a red background, even slight, got 1 while the ones having an azure background got -0.25. All the other ones, instead, got 0.

In a few cases, the explainer did not use tokens but instead worked at word-level, such as happened in KernelSHAP. For these cases, in order to calculate accurately the correlation, the assigned score has been distributed to all the tokens composing the word.

The scores for each explainers have been inserted in a matrix where the columns were reporting the 7 explainers used in this chapter, and the rows were reporting the concatenation of the token of the 10 sentences selected from the misogyny dataset. This results in a big matrix of size 7×204. After collecting and inserting all the scores for all the explainers, the Pearson correlation coefficients have been calculated between couples of vectors, where for "vector" is meant the concatenation of scores for all the tokens of all the sentences considering a single explainer.

$$\text{corr}(A, B) = \frac{\text{cov}(A, B)}{\sigma_A \sigma_B} \tag{7.5}$$

Where $\text{cov}(A, B)$ is the covariance, which is calculated as:

$$\text{cov}(A, B) = \mathbb{E}\left[(A - \mu_A)\,(B - \mu_B)\right] \tag{7.6}$$

The calculation of the Pearson correlation coefficient has been repeated for each couple of explainers, and the result is a symmetrical 7×7 correlation matrix, shown in figure 7.38. Analyzing the correlation matrix is evident the correlation between attention maps and effective attention maps is consistent. This finding had already

	Attention maps	Eff. attention maps	HTA	DeepSHAP	KernelSHAP	MiCE	SOC
Attention maps	1.0000	0.9460	0.2743	0.0055	0.1198	0.0863	0.1891
Eff. attention maps	0.9460	1.0000	0.2743	0.1581	-0.1000	0.2202	0.2374
HTA	0.2743	0.2743	1.0000	0.0055	0.0405	0.2192	0.1727
DeepSHAP	0.0055	0.1581	0.0055	1.0000	0.1198	0.0103	0.0111
KernelSHAP	0.1198	-0.1000	0.0405	0.1198	1.0000	0.0863	-0.0174
MiCE	0.0863	0.2202	0.2192	0.0103	0.0863	1.0000	0.1891
SOC	0.1891	0.2374	0.1727	0.0111	-0.0174	0.1891	1.0000

Figure 7.38

emerged in the qualitative analysis proposed in the previous section, where the fact that in almost all the cases attention maps and effective attention maps were leading to the same pattern in the plot of the weights. This result makes it still more evident and out of doubts that effective attention maps can be productively used only when handling longer sentences, which are more common in other types of tasks that are not classification, thus not the task analysed in this research. For this reason, it is hard to believe that using attention maps and effective attention maps together can be in any way useful, and their combined use is not encouraged.

It is considerable to notice that there is not a correlation between the two SHAP variants: DeepSHAP and KernelSHAP. Their Pearson coefficient is equal to 0.1198, which corresponds to a slight correlation but it is not significant. Also, the HTA and the DeepSHAP correlation index is lower than expected, and this is not for granted since they are both methods based on gradient attribution. The result can be interpreted as the fact that the use that they make of the gradient information is too different to lead to the same result.

The only two negative scores have been reached by the couples "KernelSHAP - Effective attention maps" and "KernelSHAP - SOC", which scored respectively -0.100 and -0.174. Even if negative, the score is too close to zero to state that there is a negative correlation between explainers.

In conclusion, this research showed that different explainers can be efficiently used together and they will lead to different explanations, that are not better or worse with respect to each other. This piece of research did not aim to ascertain the quality of an explainer. Two explainers can result to be not correlated as a result of the fact that one is high quality and the other one is bad quality, or as the result of the fact that they are both high quality but they are representing two different types of explanation. The only exception that reported a high correlation is the couple "attention maps - effective attention maps". All the other explanation algorithms are not correlated nor redundant, and they can be used altogether.

Also this research presented limitations, given by the fact that only one dataset has been considered and a restricted amount of sentences has been used. All the scores have been manually annotated and this prevented the scale of the research to higher amounts of sentences. The exploration of novel methods for the annotation

of the explanations is encouraged.

Chapter 8

Conclusions and further steps

The experimental part of this thesis is composed of three different parts. In the first part, an overview of the different explainers was proposed. After selecting ten sentences, seven false positives, and three false negatives, the output of the different explainers was analyzed and compared to gain meaningful insights into the bias and the reason for the neural network error.

Seven different explanation algorithms were analyzed: Attention Maps, Effective Attention Maps, SOC, HTA, DeepSHAP, KernelSHAP, and MiCE. Most of the explanatory algorithms used were able to capture the bias and show it to the user in the form of a group of tokens that were relevant enough, in contrast to the overall context, to drive the decision for the positive or negative class. Attention maps have proven best for early detection during the development time, while SOC, HTA, and MiCE have demonstrated their potential and ability to show how a decision was made, altogether with a clear output on which decisions about bias mitigation can be based. In the second part, the focus returned to the attention maps, and an analysis was conducted to confirm the hypothesis that vertical patterns are more prevalent in false-positive samples. The analysis results show that the hypothesis is confirmed, but the correlation is loose and cannot be used for the early detection of bias based on the number of vertical patterns alone. Finally, in the third part, the correlation between the used explainers was investigated. The goal of this experiment was to check whether two or more of the explainers are correlated, i.e., express the same explanation and whether they are redundant. The only two explainers that have been shown to be closely correlated are the attention maps and the effective attention maps. The other explainers, on the other hand, are only slightly correlated or not correlated at all and can therefore be used complementarily to investigate different aspects of the neural network.

Analyzing the ten sentences and seeing how the output of the explainers was analyzed along with the other sentences that were and were not correctly classified during the experiments, altogether with the ones which are not reported in this thesis, it becomes clear how the words referring to the female gender, as *woman/women*, *female*, adjectives such as *hysterical*, *slut*, *bitch*, *pussy*, places such as *kitchen*, *desert* and words, used in toxic contexts, such as *rape*, are most likely to capitalize the decision and be considered an unintended bias for the neural network in the misogyny context. Instead, the words *immigrant*, *rapefugee*, and *#buildthewall* had the greatest impact in the hate speech against immigrants context.

Although focusing on these words can lead to overall good accuracy in the classification task, as shown by the case in the example that achieved over 72% accuracy, the false positives emerging from the unintended bias cannot be considered acceptable and could be a cause for serious concern, especially if the unintended bias is caused by neutral words such as "woman" or "kitchen" that are used in a positive context in most cases.

The reason why these neutral words are considered critical for determining the positive class may be due to the dataset used, which does not contain enough similar examples that refer to the male counterpart. Given the good performance achieved by language models such as *word2vec*, it is suggested to investigate the impact that the dataset augmentation may have on the unintended bias, which is then reflected in the false positive rate and the false-negative rate.

Indeed, when the dataset is augmented, even artificially, it is hypothesized that the neural network does not recognize any more specific words as belonging only to a single context, which could have a positive effect on accuracy and recall. This analysis is left as a further investigation and it is not addressed in this paper. It is also important to consider that most of the unintended biases are expressed as a focus on a single token or on a limited set of tokens. This is particularly evident when analyzing the attention maps and ascertaining that often the attention is not uniformly distributed among tokens, but tends to form vertical patterns, especially for the false positives as demonstrated in the second part of this thesis.

For this reason, further potential for improvement can be found in the mitigation of the attention peaks through regularization during the training. This regularisation can be expressed as a penalization for the peaks, which should lead to flatter attention maps and fewer unintended biases. Exploration of this hypothesis is left as a further improvement.

Bibliography

[1] Yonghui Wu et al. «Google's neural machine translation system: Bridging the gap between human and machine translation». In: *arXiv preprint arXiv:1609.08144* (2016) (cit. on p. 2).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. «Neural machine translation by jointly learning to align and translate». In: *arXiv preprint arXiv:1409.0473* (2014) (cit. on p. 9).

[3] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. «Effective approaches to attention-based neural machine translation». In: *arXiv preprint arXiv:1508.04025* (2015) (cit. on p. 9).

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 10).

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «Bert: Pre-training of deep bidirectional transformers for language understanding». In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on p. 13).

[6] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. «Deberta: Decoding-enhanced bert with disentangled attention». In: *arXiv preprint arXiv:2006.03654* (2020) (cit. on p. 13).

[7] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. *Deep contextualized word representations*. 2018. URL: `http://arxiv.org/abs/1802.05365` (cit. on p. 14).

[8] E Burns. «In-Depth Guide to Machine Learning in the Enterprise». In: *Techtarget. March* (2021) (cit. on p. 17).

[9] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. «Man is to computer programmer as woman is to homemaker? debiasing word embeddings». In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 17).

[10] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. *Machine bias: There's software used across the country to predict future criminals. And it's biased against blacks. ProPublica, May 23.* 2016 (cit. on p. 17).

[11] William Warner and Julia Hirschberg. «Detecting hate speech on the world wide web». In: *Proceedings of the second workshop on language in social media.* 2012, pp. 19–26 (cit. on p. 17).

[12] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. «Measuring and mitigating unintended bias in text classification». In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society.* 2018, pp. 67–73 (cit. on p. 17).

[13] Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. «Detection of abusive language: the problem of biased datasets». In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* 2019, pp. 602–608 (cit. on p. 18).

[14] Aymé Arango, Jorge Pérez, and Barbara Poblete. «Hate speech detection is not as easy as you may think: A closer look at model validation». In: *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval.* 2019, pp. 45–54 (cit. on p. 18).

[15] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. «Hatexplain: A benchmark dataset for explainable hate speech detection». In: *arXiv preprint arXiv:2012.10289* (2020) (cit. on p. 18).

[16] Vivian Lai, Jon Z Cai, and Chenhao Tan. «Many faces of feature importance: Comparing built-in and post-hoc feature importance in text classification». In: *arXiv preprint arXiv:1910.08534* (2019) (cit. on p. 19).

[17] Piyawat Lertvittayakumjorn and Francesca Toni. «Human-grounded evaluations of explanation methods for text classification». In: *arXiv preprint arXiv:1908.11355* (2019) (cit. on p. 21).

[18] Sarthak Jain and Byron C Wallace. «Attention is not explanation». In: *arXiv preprint arXiv:1902.10186* (2019) (cit. on p. 23).

[19] Sarah Wiegreffe and Yuval Pinter. «Attention is not not explanation». In: *arXiv preprint arXiv:1908.04626* (2019) (cit. on p. 23).

[20] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. «Revealing the dark secrets of BERT». In: *arXiv preprint arXiv:1908.08593* (2019) (cit. on p. 24).

[21] Greg Ridgeway, David Madigan, Thomas Richardson, and John O'Kane. «Interpretable Boosted Naıve Bayes Classification.» In: *KDD*. 1998, pp. 101–104 (cit. on p. 27).

[22] Cynthia Rudin. «Please stop explaining black box models for high stakes decisions». In: *Stat* 1050 (2018), p. 26 (cit. on p. 28).

[23] Tao Lei, Regina Barzilay, and Tommi Jaakkola. «Rationalizing Neural Predictions». In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 107–117. DOI: 10.18653/v1/D16-1011. URL: https://aclanthology.org/D16-1011 (cit. on p. 28).

[24] Christoph Molnar. *Taxonomy of Interpretability Methods*. christophm.github.io. URL: https://christophm.github.io/interpretable-ml-book/taxonomy-of-interpretability-methods.html (cit. on p. 29).

[25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. «" Why should i trust you?" Explaining the predictions of any classifier». In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144 (cit. on p. 30).

[26] Scott M Lundberg and Su-In Lee. «A unified approach to interpreting model predictions». In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 30).

[27] Alexis Ross, Ana Marasović, and Matthew E Peters. «Explaining nlp models via minimal contrastive editing (mice)». In: *arXiv preprint arXiv:2012.13985* (2020) (cit. on p. 33).

[28] Xisen Jin, Zhongyu Wei, Junyi Du, Xiangyang Xue, and Xiang Ren. «Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models». In: *arXiv preprint arXiv:1911.06194* (2019) (cit. on p. 33).

[29] Damian Pascual, Gino Brunner, and Roger Wattenhofer. «Telling BERT's full story: from Local Attention to Global Aggregation». In: *arXiv preprint arXiv:2004.05916* (2020) (cit. on p. 51).

[30] Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. «On identifiability in transformers». In: *arXiv preprint arXiv:1908.04211* (2019) (cit. on pp. 57, 60, 64).

[31] Chandan Singh, W James Murdoch, and Bin Yu. «Hierarchical interpretations for neural network predictions». In: *arXiv preprint arXiv:1806.05337* (2018) (cit. on p. 61).

[32] Brendan Kennedy*, Xisen Jin*, Aida Mostafazadeh Davani, Morteza Dehghani, and Xiang Ren. «Contextualizing Hate Speech Classifiers with Post-hoc Explanation». In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. to appear (cit. on p. 61).