Master thesis Artificial Intelligence

Explainability of Transformers for Authorship Attribution

Ivan Kondyurin

9086765

First supervisor: Dr. Denis Paperno

Second examiner: Yupei Du

Graduate School of Natural Sciences
Utrecht University
Netherlands

July 29, 2022

Utrecht University

Contents

Abstract **6**

1. Introduction **7**

1.1 Research question 8

1.2 Structure 11

2. Authorship attribution methods **11**

2.1 Traditional approaches to attribution 11

2.2 Quantitative approaches to attribution 12

2.2.1 Single-measurement features 12

2.2.2 Feature selection and Complex features 16

2.3 Modern attribution methods 16

2.3.1 Profile-based supervised methods: probabilistic and compression models 16

2.3.2 Instance-based methods 17

2.4 Authorship attribution datasets 20

2.4.1. Traditional datasets 20

2.4.2 Modern cross-domain datasets 20

2.4.3 Datasets discussion 21

3. Transformer models in authorship attribution **22**

3.1 Transformer overview 23

3.2 Transformer models fine-tuning 24

3.2.1 BERT 25

3.2.2 Transformer Language Models evolution 26

3.3 Enhanced transformers 26

3.3.1 Computational cost 26

3.3.2 Improved architectures 27

3.4 Transformers application for text classification 28

3.4.1 Transformers for authorship attribution 28

4. Transformer explainability 30

4.1 Defining explainability 30

4.2 Attention weights as an explainability tool 31

4.2.1 Critical view at attention explainability 32

4.2.2 Justification of attention explainability 33

4.2.3 Attention explanations in NLP transformers 34

4.2.4 Transformer explanations beyond attention 35

5. Model for authorship verification 35

5.1 Model task overview 35

5.2 The PAN fanfiction dataset 36

5.3 The model implementation 38

5.3.1 Recent approaches to Transformer-based AA 38

5.3.2 BERT segment classification model 40

5.3.3 BERT segment classification implementation 41

5.3.4 Obtaining embeddings 45

5.3.5 Final layer classifier 45

5.4 Model results 45

5.5 Model discussion 46

6. Final classifier interpretation 47

6.1 LIME-tabular explanations overview 47

6.2 LIME-tabular explanations analysis 49

6.3 CLS embeddings analysis 50

6.4 Comparison of LIME-tabular and difference-based features 52

6.5 Untuned CLS embeddings analysis 53

6.6 Correlation between CLS embeddings and classifier nodes 55

6.7 Classifier layer discussion 57

7. LIME explanations for segments 57

7.1 LIME overview 58

7.2 LIME-text for attribution explanations 58

7.2.1 Class 0 explanations 59

7.2.2 Class 1 explanations 60

7.2.3 Analysis of explanation difference 61

7.3 Pair-aware perturbations in LIME 62

7.3.1 Non-bag-of-words perturbation 62

7.3.2 Bag-of-words perturbation 63

7.4 Feature extraction with pair-aware perturbations 63

7.4.1 Extracted features overview 65

7.4.2 Relations between LIME estimations and prediction logits 65

7.4.3 Co-occurrence of important features 66

7.4.4 Adversarial examples 67

7.4.5 Feature importance discussion 70

8. Attention analysis 71

8.1 CLS attention overview 71

8.1.1 Preliminary per-layer analysis of CLS attention 72

8.1.2 Discussion of the general CLS behavior 73

8.2 Attention visualization overview 73

8.2.1 Averaged attention visualization 74

8.2.2 Per-layer attention visualization 75

8.3 CLS attention as means of retrieving tokens relevant for classification 82

8.3.1 Distribution of relevant attention with respect to CLS 82

8.3.2 Selective coverage in CLS 83

8.3.3 Types of tokens attended by CLS 84

8.4 Highly attended tokens as linguistic features 85

8.4.1 Selectivity and causal interpretation 88

8.5 Comparison with other known features 88

8.5.1 Overlap with features extracted by Boenninghoff et al. 88

8.5.2 Overlap with features commonly used in authorship attribution 90

8.6 Overlap with LIME 92

8.6.1 Feature comparison 92

8.6.2 Feature importance 94

8.6.3 Label inversion 96

8.7 Names as features 96

8.7.1 Overview 96

8.7.2 Names ablation and alteration techniques 97

8.7.3 Names ablation and alteration results 98

9. Conclusion 101

9.1 Discussion of research question Q1 103

9.2 Discussion of research question Q2 104

9.3 Future work 106

Bibliography 108

Appendix 118

A. Description of all CLS-related attention 118

Abstract

Authorship attribution attempts to establish the author of a particular text. In this work, we examine the capabilities of transformer-based models in the subtype of attribution task referred to as authorship verification, which involves determining whether the texts are created by the same author. A few works have been suggested that applied fine-tuned Transformer models in this field. Such approach is motivated by their excellent performance and adaptability (fine-tuning can be performed on texts of different sizes and genres, and different pre-trained model checkpoints enable switching between languages). However, they are not as transparent as the traditional methods, in which features that quantify the style (stylometric features) are selected to maximize the distance between texts. To tackle this problem, we first implement a model for authorship verification based on BERT architecture and then investigate the way its predictions are made by applying an adapted LIME explainer and proposing an attention-based relevant feature extracting procedure. We then compare the two approaches and analyze their explainability from the causal perspective by input ablation and alteration to verify that they can retrieve the features that have a strong influence on the model predictions. We also describe and classify the extracted features from a linguistic perspective.

1. Introduction

Authorship attribution (AA) is a traditional field of philological and linguistic studies. In general case, it attempts to establish the person who wrote a particular text. However, a variety of subtasks motivated by real-world problems exists, including authorship verification, which aims at proving that a certain author has indeed created a given piece of text (Kestemont et al., 2020), attribution of texts with an open or closed set of candidate authors (Stamatatos et al., 2017), authorship obfuscation (Barlas and Stamatatos, 2020), and more.

Traditionally, researchers in authorship attribution relied on data from extralinguistic sources, such as biographical and historical information and physical evidence (Stamatatos, 2009). However, in many cases, such information was unavailable or insufficient to differentiate between several closely related authors that, for example, worked for the same newspapers and published topically related articles (Holmes, 1994; Glaudes et al., 2019). In such cases, statistical analysis of stylistic features was used. For each case of disputable authorship, a feature-based description of each author's texts was performed, sometimes by manual labeling (Marusenko, 1990), and a set of statistically significant parameters that could successfully discriminate between texts of known authorship was selected.

The set of exploited features differed considerably across studies. A detailed historical review of the development and refinement of potential features is given by Juola (2006), while a practical comparison of the effectiveness of available parameters drawn from different levels of language can be found in Stamatatos (2009) and Sari et al. (2018). In general, many basic word-level features were shown to be ineffective, while frequencies of the word and character n-grams indeed turned out to be capable of reflecting individual stylistic preferences (Burrows, 2002; Eder et al., 2016). More complex and high-level features, such as sentence structure or frequency of rewrite rules were also used (Marusenko, 1990; Stamatatos, 2009; Sari et al., 2018), but the complexity of automatic extraction made them less widely exploited, particularly for large texts.

Although statistical methods are still widely used, they can be insufficient in some cases. Firstly, the accuracy of statistical predictions in some languages may decline for shorter texts (Eder et al., 2016; Rybicki and Eder, 2011; Rybicki and Eder, 2013; Hirst and Feiguina, 2007), as the feature frequency distribution can be distorted due to the limited size. For example, Burrows' Delta, a widespread statistical metric for discriminating between text styles based on word frequency distribution (Eder et al., 2016, Evert et al., 2017; Glaudes et al., 2019), was only proposed by Burrows (2002) for texts "exceeding about 1500 words in length" and can only be used for reducing the a-priori set of authors if the texts are between 100 and 1500 words in length. Despite various improvements suggested to this method (Hoover, 2004; Hoover, 2005; Burrows, 2007; Argamon, 2008; Jannidis et al., 2015; Smith and Aldridge, 2017), the general constraint persisted and was considered a significant limitation for attribution of texts with limited length.

Moreover, stylistic differences across genres may dominate over those of different authors (Juola, 2006), which impairs the value of statistical feature analysis for cross-domain attribution, in which sufficient data for a given author is only available in one genre, while attribution needs to be performed on text of a different genre. If the set of candidate authors is open, that is, the true author of the text was not included in the training dataset, traditional selection of parameters is also harder since we cannot ensure their statistical significance for a particular pair of authors.

These considerations motivated the use of neural network-based models for authorship attribution. Over the last years, different architectures have been used (Zheng, 2006; Khosmood, 2006; Barlas and Stamatatos, 2021). Among those showing the best performance were RNNs, CNN-based models which accounted for peculiarities of punctuation, Siamese Networks trained for text comparison, and various hybrid architectures and ensembles of networks (Stamatatos et al., 2018; Kestemont et al., 2020). Comparatively modern Transformer models (Vaswani et al., 2017), which are capable of grasping long-distance relations to accomplish various NLP tasks, have also been recently applied to authorship attribution (Ordoñez et al., 2020; Fabien et al., 2020; Peng et al., 2021).

However, no consensus has been reached regarding the optimal architecture and, more generally, the ability of fine-tuned Transformer language models to learn stylistic specificities and use them for authorship prediction better or on a par with other contemporary models. While some solutions surveyed by Barlas and Stamatatos (2020) and BertAA suggested by Fabien et al. (2020) showed state-of-the-art performance on benchmark datasets, other ones failed to beat the baselines. The performance of Transformer models during the PAN20 and PAN21 competitions was also ambiguous: the model by Ordoñez et al. (2020) performed efficiently on the first testing dataset but showed a significant decline on the second one (Kestemont et al., 2020), while the model of Peng et al. (2021) has demonstrated near state-of-the-art results.

Apart from the matter of performance, another serious concern with Transformers for AA, as with other deep learning models, is the explainability of the output. Since the result of attribution may have considerable consequences in real life, including reconsideration of authorship as such as well as a potential accusation of plagiarism and issues with royalties. Therefore, the possibility of explaining and motivating the decision is highly desirable.

Although statistical methods are weaker in some subfields, they allow the researchers to extract important features and explicitly quantify the difference between texts based on their values, which fully interprets the procedure. Machine learning models cannot provide such high level of interpretability, but reliable explainability techniques also exist for some types of models (Jain and Wallace, 2019). For Transformers, on the contrary, the discussion of which techniques are most appropriate for extracting meaningful causal relations between input and output is still in progress (o'Riedl, 2019).

The task of Transformer-based authorship attribution is therefore not only in creating a model with sufficient prediction accuracy but also in verifying that the model learns information meaningful for this task, which is the stylistic footprint of the author, and not some irrelevant information, such as topic- or genre-specific traits. To that end, available explainability methods, both model-agnostic and specific for Transformer architecture, need to be applied to our model, and their output needs to be analyzed to check which features the model uses and if they can be treated as stylistic.

1.1 Research question

The general idea of the current work is to explore whether the use of Transformer models for authorship attribution is feasible and, if so, to which extent they are interpretable in the course of such a task — in particular, whether it is possible to detect which features play the most important role. If the features are retrievable, our goal is also to analyze and classify them with respect to the existing inventory of parameters used for style analysis.

Therefore, our goal in this research is not only to build a model that can solve the problem of authorship attribution utilizing a pre-trained transformer-based model but also to analyze how the predictions are made. To that end, we scrutinize different explanation techniques and analyze their relationship with the input characteristics.

The **relevance** of this problem is justified by the growing popularity of fine-tuned or few-shot-learned Transformer models, which leads to their application in various fields, even sensitive ones. The remarkable performance of these models in other tasks may persuade the audience that their output needs to be trusted without additional investigation and consideration. Therefore, given the importance of potential negative consequences for erroneous results in AA, the explainability of the model output becomes crucial.

We suggest that the application of AA models to the domain of texts written by independent creators needs to be done with particular caution. The rapidly increasing number of created texts encourages automatization of plagiarism and fraud detection, but for independent authors that use social media and other online platforms as their primary way of publishing, unfair treatment due to mistakes in classifications may result in financial and reputational losses. One of the fields in which such problem seems to be relevant is fanfiction, i.e., literary texts created by non-professional writers "in the tradition of a specific cultural domain" (Stamatatos et al., 2018; Kestemont et al., 2020). Niche fanfiction writers often work independently without formal evidence of their authorship or even publish anonymously, which puts them at risk when the author needs to be established. We emphasize that the ability to motivate the model's decision even for accurate classifications is highly desirable, and for the Transformer models explainability is still among the major challenging aspects.

Besides, we highlight that the explanations need to be task-specific and created with the end user to whom they are directed in mind. For the task in question, such users are primarily experts, the linguists performing attributional studies. Therefore, the model should, ideally, be able to reveal the most important linguistic features that are used to represent stylistic differences, and such linguistic features should be meaningful in terms of style analysis. Thus, for the use of Transformers to be justified in AA, we expect such models to fulfill two **requirements**:

(1) *efficiency:* showing competitive performance compared to other NN models and outperforming the common baselines

(2) *task-specific explainability:* allowing for the extraction of important features that are meaningful, correspond to some existing stylometric features of any level, and are indeed discriminative for this model in the current AA task.

These requirements are reflected in the structure of our **research question**, which is subdivided into the following parts:

Q1
Is it reasonable to utilize pre-trained Transformer models for the task of authorship attribution, or their performance does not surpass that of smaller models?
Q1.1
How does the ability to process longer sequences contribute to the Transformer performance? Is it required for proper attribution to be able to process a longer sequence simultaneously?

Q2-1
If the answer to Q1 is True, *which meaningful parameters, if any, can we extract, and by which means (for example, by analyzing attention matrices, or permuting input, or applying existing model-agnostic methods)?*
Q2-1.1
Do any of these patterns correspond to stylometric features used in the traditional stylistic analysis for attribution?

Q2-2
If the answer to Q1 is False, *what is the cause of this failure? Is it specific to our proposed solution, related to the known bottlenecks of current Transformer models, or caused by some fundamental limitations of the Transformer architecture?*

Question **Q2-1** corresponds with six possible hypotheses:

Hypothesis I implies that explanations related to existing stylometric features may be found by visualizing the attention weights of different heads at different levels
Hypothesis II aims at finding valuable features by generating explanations on fully connected layers at different levels
Hypothesis III suggests that the uppermost layer, the classifier itself, can provide sufficient explanations
Hypothesis IV involves explanations based on complex features (e.g. from combining multiple attention heads) observed, among other methods, using input permutation
According to *Hypothesis V*, explanations can be generated by combining information from the sources mentioned above
Hypothesis VI accounts for the negative answer, according to which no meaningful features that reveal causal relations between input and output and correspond to any of the existing stylometric features could be found

If the answer is False (and the Transformer model could be trained in principle but did not outperform the baselines), **Q2-2** is used to determine the possible causes of this issue:

Is this insufficient performance specific to our proposed solution, related to the known bottlenecks of current Transformer models, or caused by some fundamental limitations of the Transformer architecture?

In this case, four hypotheses are in place:
Hypothesis I suggests that poor performance is due to the limited input length of existing Transformer models that does not allow them to process the text as a whole and learn long-distance stylistic patterns; therefore baseline models with limited input window size would also show a corresponding decrease in performance
Hypothesis II implies that the size of the pre-trained Transformer language model, i.e., the number of parameters or hyperparameters, constitutes a limitation, and a larger model (for example, a large version of GPT-J as opposed to a small one) could perform better in this task
Hypothesis III states that the general Transformer architecture is the reason for insufficient performance as it is incapable of properly learning quantitative features that are important for AA, and a better performance can be expected from a model that makes use of different architecture for the language model, such as LSTM-based one
Hypothesis IV covers the negative scenario in which none of these expected justifications can be proven.

The anticipated contributions of our work therefore include:

- a model for verifying the authorship of a pair of texts
- a proposed procedure for explaining them

1.2 Structure

In the course of answering these research questions, we will first provide theoretical motivation and literature review for the problems and postulates outlined above and then describe the stages of developing our proposed solution.

This work is **structured** in the following way:

- Firstly, a historical overview of existing methods of AA is provided in **Chapter 2**, including traditional philological analysis, statistical stylometry, and machine learning techniques
- After that, in **Chapter 3** we provide a more in-depth survey of the Transformer architecture with an emphasis on its application to text classification and AA in particular. Particular attention is paid to known weak points of the existing solutions, such as limited input sequence length
- In the following **Chapter 4**, the interpretability of transformers is discussed, with a focus on the explanation potential of attention weights given their model-specific nature. Alternative ways of generating explanations are also outlined
- We start the practical part with **Chapter 5** by describing the model architecture, reflecting on its implementation, and evaluating its performance with different input sizes
- The interpretation of the model starts with a detailed investigation of the final classification layer which averages predictions in multiple segments in **Chapter 6**. We utilize feature extraction and analyzer the input embeddings to motivate why the classifier can be disregarded for the future explanation
- In **Chapter 7**, we proceed with using LIME-text explainer to obtain most important features, analyze their statistics and perform ablation experiments to verify their importance and reveal dependencies
- Finally, in **Chapter 8** we analyze the attention in all heads and layers of the BERT component to highlight the most relevant types of relations and use the relevant attention matrices to extract the most highly attended tokens. We compare these tokens with LIME features assess their importance. In the end, we provide an additional investigation of names used as features, given their frequent occurrence as the most important ones.

2. Authorship attribution methods

A detailed survey of stylometry development can be found in Holmes (1994). Juola (2006) provides a comprehensive overview of the field with a section dedicated to a critical account of various attribution techniques from a modern perspective.

2.1 Traditional approaches to attribution

In traditional approaches, the attribution was largely performed based on the evidence from external sources (such as biographical data, incipits, and colophons (Stamatatos, 2009)), internal data (self-references, topical and ideological homogeneity), bibliographical evidence, historical facts, and physical evidence (the analysis of ink, handwriting, watermarks).

One of the earliest accounts of authorship can be found in St. Jerome (Hulley, 1944) with respect to the critical study of Biblical texts. Among the criteria for questioning the originality of a passage of text, he considers the contradiction between the ideas in the text and the author's doctrine, the inclusion of quotes or references to events that occurred after the author's death or were unknown to the author. St. Jerome also pays attention to the shift in style, including the occurrence of words, collocations, and expressions atypical for texts of certain author. This can be considered an early example of stylistic features analysis.

2.2 Quantitative approaches to attribution

In some cases, authorship may be ascertained based solely on the historical and biographical data along with high-level philological analysis.

However, when these knowledge sources are insufficient (Glaudes et al., 2019; Marusenko, 1990), a more detailed account of the texts at issue is required. Such procedure of authorship attribution generally relies on stylometry, which includes creation, formal representation, and comparison of stylistic fingerprints of authors (Holmes, 1994; Holmes, 1998; van Halteren, 2005).

The first evidence of applying quantitative methods to formally describe the writing style dates back to the late 19th century when Mendenhall (1887) attempted to determine the authorship of plays officially attributed to Shakespeare. Among other early attempts is the work by Lutoslawski (1898) who used stylometric features to attribute a number of Plato works. These works were then followed by influential statistical studies by Zipf (1932), Yule (1938; 1944), and Simpson (1949).

According to Holmes (1994), the research goal with respect to the feature selection was determined as follows: "The stylometrist therefore looks for a unit of counting which translates accurately the 'style' of the text, where we may define 'style' as a set of measurable patterns which may be unique to an author". The underlying assumption made here is that the style of a single author is considered constant for the text corpus in question and that this style necessarily differs from that of other authors. The style in this paradigm is therefore viewed as the unique fingerprint of an author, or individual "stylome".

The exact set of methods depends on the type of research question (Juola, 2006): compared to closed-set attribution, open-set attribution needs to rely more on the exact distance between the test set documents and all of the a priori classes rather than on finding the nearest a priori class (Eder et al., 2016). Authorship verification is treated with a stronger emphasis on the pairwise similarities, which are currently often processed using Siamese networks (Tyo et al., 2021). For author profiling, establishing the author class is insufficient: this class itself is also to be described with some attributes, such as age, education, native language (Barlas and Stamatatos, 2021).

For this work, we will focus on the verification of authorship.

2.2.1 Single-measurement features

According to (Neal et al., 2017), there is still no general consensus on the optimal feature set. In general, features frequently exploited for quantifying the style in authorship attribution involve vocabulary, syntax, semantics, and characters (Stamatatos, 2009). The overview of such features together with the tools required for extracting them, as formulated by Stamatatos, are presented in Table 2.1.

Features		Required tools and resources
Lexical	Token-based (word length, sentence length, etc.)	Tokenizer, [Sentence splitter]
	Vocabulary richness	Tokenizer
	Word frequencies	Tokenizer, [Stemmer, Lemmatizer]
	Word *n*-grams	Tokenizer
	Errors	Tokenizer, Orthographic spell checker
Character	Character types (letters, digits, etc.)	Character dictionary
	Character *n*-grams (fixed length)	–
	Character *n*-grams (variable length)	Feature selector
	Compression methods	Text compression tool
Syntactic	Part-of-speech (POS)	Tokenizer, Sentence splitter, POS tagger
	Chunks	Tokenizer, Sentence splitter, [POS tagger], Text chunker
	Sentence and phrase structure	Tokenizer, Sentence splitter, POS tagger, Text chunker, Partial parser
	Rewrite rules frequencies	Tokenizer, Sentence splitter, POS tagger, Text chunker, Full parser
	Errors	Tokenizer, Sentence splitter, Syntactic spell checker
Semantic	Synonyms	Tokenizer, [POS tagger], Thesaurus
	Semantic dependencies	Tokenizer, Sentence splitter, POS tagger, Text Chunker, Partial parser, Semantic parser
	Functional	Tokenizer, Sentence splitter, POS tagger, Specialized dictionaries
Application-specific	Structural	HTML parser, Specialized parsers
	Content-specific	Tokenizer, [Stemmer, Lemmatizer], Specialized dictionaries
	Language-specific	Tokenizer, [Stemmer, Lemmatizer], Specialized dictionaries

Table 2.1: *Types of stylometric features (from Stamatatos (2009))*

Vocabulary features

Plain linguistic features that are extracted by means of a single measurement served as the starting point for formal stylometry. Holmes (1994) traces the earliest formal research of vocabulary to the aforementioned Mendenhall (1887) who suggested using word lengths as a distinctive feature to determine the writer.

Multiple lexical features have been assessed as potentially meaningful for authorship attribution, including average length of words and sentences, average count of syllables per word, POS distribution, type/token ratio, and measures of vocabulary richness (Stamatatos, 2009). Various vocabulary richness functions that aim at quantifying the diversity of lexicon used in a text are type-token ratio V/N (vocabulary/number of words), the number of hapax legomena (that is, words only occurring once) (de Vel et al., 2001) and sometimes also dis legomena, tris legomena, and so on (words with double and triple occurrences respectively). However, the values of these parameters heavily depend on the text length, and therefore a number of functions for normalized counts of lexicon diversity were proposed, such as Yule's K (Yule, 1944; Tanaka-Ishii and Aihara, 2015) and Simpson's D index (Simpson, 1949). The use of vocabulary richness for AA was criticized in (Hoover, 2003). A broader review of criticism towards this approach can be found in (Juola, 2006).

However, some plain single-measurement features yielded promising results in the attribution of specific text corpora. One such case is the use of letter counts (Merriam, 1998) which turned out to be unexpectedly efficient for the case of Shakespeare vs. Marlowe authorship question. Merriam claimed that "of counting the letters in the 43 plays was the implausible discovery that the letter 'o' differentiates Marlowe and Shakespeare plays to an extent well in excess of chance" and established a threshold value for 'o' frequency that enabled to attribute the set of plays in question to the Shakespeare class or to that of Marlowe.

According to Juola (2006), a possible reason for the relatively poor performance of the aforementioned features is that they were selected manually before the actual analysis of the data, based on their expected contribution to the style characteristic. A more promising approach can be to reveal the distinctive features from the data analysis by determining the regular, noticeable, and explainable differences between text sets. Such approach is sometimes referred to as proper stylometric analysis (Boenninghoff et al., 2019). Within this paradigm, new methods were introduced, such as the analysis of synonym pairs. However, it was not applicable to some datasets due to the data sparsity problem since the number of strict synonyms is severely limited.

To avoid the sparsity issue, Mosteller and Wallace (1964) suggested focusing on function words (articles, prepositions, conjunctions) that carry little lexical meaning but define syntactic or semantic functions. While in many other areas of language processing, such as sentiment analysis or topic modeling, these words are commonly removed beforehand, in authorship attribution they allowed achieving positive results in attributing the Federalist papers. The reason for such success is that function words are topic-independent, which reduces the influence of topical differences across authors, and relatively interchangeable, which ensures that different authors could freely alter the way of expressing themselves according to personal stylistic preferences. The authors used Bayesian statistical analysis of frequency distributions of the small set of function words. Stamatatos (2009) marks this study as the one initiating "nontraditional authorship attribution" that relies on simple quantitative parameters that can easily be extracted automatically, as opposed to traditional attribution that relied on human experts labeling the data with complex parameters and selecting statistically significant ones.

The Federalist Papers (a set of 85 essays written under the pseudonym *Publius*) have been an important target for AA since then and are now seen as a benchmark. This is a closed-set attribution problem that is currently considered solved, and the first proposed solution is described as "the most famous and widely cited statistical analysis of authorship" (Juola, 2006). Mosteller and Wallace analyzed the relative frequencies of 30 function words, and since then many works relied on a similar technique.

Currently, a common lexical feature is the (relative) frequency of a number of most frequent words or n-grams with substantial variation in their number, selection conditions, and preprocessing (Burrows, 2002; Eder et al., 2016; Sari et al., 2018). However, the raw counts of these feature values are not used alone: instead, various distance metrics or simple statistics such as PCA can be applied. Even though the unfiltered list of most frequent words would typically include function words, many researchers still use a specifically defined set of function words instead. Among them are Argamon et al. (2007) with a list of 675 words, and Koppel and Schler (2003) who used 480 words.

Syntactic features

A more high-level way of representing stylistic information is by considering syntactic features, with the assumption being that an individual style contains specific syntactic patterns more frequently than other styles, and this frequency is to a certain degree consistent among different texts. The importance of function words for attribution provides additional evidence for this view. The downside of including diverse and complex syntactic features is the increasing complexity of labeling. Without reliable techniques for automatic parsing, a substantial amount of human expert work was needed to count the feature values. Automatic syntactic analysis, on the other hand, could lead to multiple errors in parsing that may skew the feature counts.

Features counted manually or semi-automatically (i.e., using guided rule-based systems to assist labeling) were extensively used by Marusenko (1990). He proposed a set of 56 a-priori parameters, most of them being syntactic, of which those were chosen that showed statistical significance (t-test values above 1.96) in discriminating texts of two authors. Among such syntactic parameters are the number of words per simple sentence, number of subordinate clauses, embedded clauses, homogenous elements, determiners, infinitive groups, and more. This method with some alterations was successfully used for a number of authorship problems, including that of Corneille-Molière (Rodionova, 2007). For this work, the set of parameters initially designed for the Russian language was adapted to better represent the stylistic capabilities of the French language. However, most of the 51 parameters used in this study correspond with the original ones.

Stamatatos (2009) traces the first attempt to employ more elaborate syntactic features for the English language to Baayen et al. (1996). They also used a semi-automatic approach for syntactical annotation to ensure that parsing is correct. To create the set of features, they extracted the frequencies of rewrite rules (that is, rules according to which a phrase is constructed out of immediate constituents). The resulting features performed better than vocabulary richness metrics and some lexical features. Gamon (2004) used rewrite rule frequencies extracted automatically and observed that in combination with lexical features they can perform better than the latter alone.

Stamatatos (2001) attempted to perform attribution of texts in the Greek language using phrase boundaries, also extracted automatically, to count the frequencies of constituents. Hirst and Feiguina (2007) used bigram frequencies of partial parsing with varying complexity.

A more simple way of incorporating additional syntactic information using automatic analysis would be to use POS tags, but they are not particularly informative outside the context. To address this, it is possible to count frequencies of n-grams (Sari et al., 2018; Koppel et al., 2009) or POS n-grams (Stamatatos, 2009). Among solutions in this direction are those proposed by Argamon-Engelson et al. (1998) and Kukushkina et al. (2001).

Semantic features

Due to the complexity and not fully reliable accuracy of automatic semantic analysis, few attempts have previously been made to exploit semantic features in AA. One such study that provides a detailed description of feature extraction and prediction results is that of Argamon et al. (2007). They used semantic information semi-automatically extracted from WordNet (Miller et al., 1990) to create features that reflected high-level semantic information associated with certain words or phrases and indicated their semantic functions, such as “elaboration” and “apposition”. The authors showed that this set of parameters combined with lexical and syntactic features may improve the classification, but the effect of those features independently was not reported.

Structural and orthographic features

Another possible addition outlined by Juola (2006) and utilized in some of his works (Juola, 2003) is the inclusion of character n-grams. They were first used by Kjell (1998) who performed successful attribution of the Federalist papers using character bi- and trigrams.

The reasons for this success are manifold. Firstly, character-level features can better account for morphologically related words that can correlate in terms of relative accuracy in texts of a particular author. Secondly, character-level analysis enables consideration of punctuation and formatting if such symbols are not removed. This can be particularly important in attributing internet “microtexts”, specifically chat messages, comments, and social media posts (Boenninghoff et al., 2019; Suman et al., 2021), in which the specific patterns of spaces, punctuation, and indentation can bear importance for AA.

Another trait of character-level features is the ability to incorporate the author’s regular misspellings and atypical uses of punctuation while also being tolerant to noise, such as sporadic typographical errors. Koppel and Schler (2003) considered regular errors important individual traits of authors. However, Juola (2006) warns against relying solely on such patterns in datasets since they may have been disturbed during the preprocessing. In the case of text layout, it can also be changed during editorial processing. Therefore, one cannot ensure that a particular structural trace was created by the author herself unless specific evidence exists.

The practical advantage of character-level n-grams compared to word-level ones is their language-independent nature and simplicity of extraction using most basic programming tools instead of language-specific tokenizers. However, they increase the dimensionality of feature space compared to word-based parameters.

2.2.2 Feature selection and Complex features

While word- and character-level n-grams may be successfully used independently, more complex features work best in combination with each other or with n-grams. However, adding all available features to the classifier also may be implausible due to increasing dimensionality which can lead to overfitting (Stamatatos, 2009).

Therefore, the matter of feature selection is important. Some straightforward techniques, such as the one used by Marusenko (1990), included calculation of Student's t-test values and choosing the features that have values above the selected threshold for a given pair of text sets. Forsyth and Holmes (1996) compared the sets of n-grams obtained by frequencies and by distinctiveness and found the latter favorable. However, in other cases, frequency-based features were shown to be more efficient than those collected using such criteria for examining discriminatory power of features as information gain (Houvardas and Stamatatos, 2006) and odds ratio (Koppel et al., 2006). Algorithmic approaches were also applied to that end: Li et al. (2006) utilized a genetic algorithm for feature reduction, and its application resulted in increased accuracy with a mere half of the initial features.

Lexical features, such as most frequent words, can show certain correlations. In the Federalist Papers, such correlation was noted for "has" and "have", as well as for "it" and "that" (Mosteller and Wallace, 1964). To avoid overweighting particular features, the reduction of highly correlated feature dimensions can be performed. A popular technique to ensure the independence of dimensions is principal components analysis (PCA) applied by Burrows (1987; 1989; 1992) and studied in detail by Sebastiani (2002). PCA aims at generating smaller ordered sets of new uncorrelated variables (principal components) that explain as much of the data variance as possible. It thus yields linear combinations of initial features. Typically, two components with the highest explainability are mapped to a two-dimensional space for easy visualization.

2.3 Modern attribution methods

2.3.1 Profile-based supervised methods: probabilistic and compression models

Stamatatos (2009) provides an interesting criterion for grouping the attribution methods. The author notices that some methods treat training data per author cumulatively, thus defining the author's profile, and the differences within the author's class are disregarded ("profile-based methods"), while others treat each instance of the training sample individually ("instance-based methods").

An example of the first type of model is a probabilistic model that extracts the profiles of candidate authors by concatenating their texts and then uses probabilistic classifiers, such as naïve Bayes, to maximize the probability of training texts belonging to their true authors' profiles based on their distance to each author's class according to a chosen distance function. Such a model was first used for AA by Mosteller and Wallace (1964), and a detailed account of probabilistic classifiers can be found in Sebastiani (2002).

Another category that uses concatenated texts to represent authors' profiles is compression models. They calculate cross-entropy between the target text and the author's profiles using compression algorithms (Marton et al., 2005). In this method, the gain in size after adding the target text to the compressed set of a candidate author can serve as a similarity measure. The compression model was used in the study by Kukushkina et al. (2001) that showed that the RAR compression algorithm performed most efficiently in this task.

Other methods of AA are considered instance-based as they treat the contribution of each training text separately. They require a better account of the training data processing: each author's class needs to be represented by multiple texts of comparable size. The length of these texts is an important hyperparameter: they need to be sufficient to represent the author's style in a consistent manner but also correspond with the average text length to avoid excessive variation. Hirst and Feiguina (2007) observed the performance of their classifier with text pieces of different length and discovered that performance significantly decreases for smaller texts (with lengths of 200 and 500 words). Similar findings were made for Burrows' Delta (Burrows, 2002), for which a recommended text size of at least 1500 words was initially claimed. Thus, large texts need to be split into chunks of equal length, while for short texts data augmentation techniques can be used (Glaudes et al., 2019).

2.3.2 Instance-based methods

When the set of authors is not known a priori, available techniques for statistical analysis include Multidimensional Scaling (MDS) (Mead, 1992) and Cluster analysis. A large number of features also necessitates the emphasis on interpretability and visualization of the feature values, as they are harder to represent. Cluster analysis and MDS involve the calculation of intertextual distances that represent the degree of dissimilarity. The difference in cluster analysis, according to (Juola, 2006), is that after measuring pairwise distances the closest pairs are grouped together and replaced with a new item that represents the cluster. This procedure repeats until a single cluster is formed, and the result can be displayed as a dendrogram with binary branching (with each split representing a pairwise combination). MDS can be used with different distance metrics, such as «linguistic cross-entropy» employed by Juola (1997).

Similarity-based models

Similarity-based models are based on the idea of pairwise similarity between the target text and all texts from the training dataset. When this similarity is calculated according to a selected metric function, the text is attributed to the most likely class that is typically found using the k-nearest neighbors algorithm (Fix and Hodges, 1951).

One of the most successful cases of combining values of multiple features to separate different classes of authors is Burrow's Delta. According to the author's definition, the Delta measure is "the mean of the absolute differences between the z-scores for a set of word-variables in a given text-group and the z-scores for the same set of word-variables in a target text" (Burrows, 2002).

Burrow's delta was introduced with the view to grasping the stylistic distance between texts. It is based on the Z-score values of m most frequent words or word-POS pair in each text. The Z-score is calculated as a difference between a relative frequency of a word and its mean frequency in the reference corpus divided by standard deviation:

$$z_i(D) = \frac{f_i(D) - \mu_i}{\sigma_i}$$

Distance is then computed as the sum of differences in Z-scores between authors D and D' for each frequent word, which is then divided by their total number:

$$\Delta_B(D, D') = \frac{1}{n} \parallel \mathrm{z}(D) - \mathrm{z}(D') \parallel_1 = \frac{1}{n}\sum_{i=1}^{n_w} |z_i(D) - z_i(D')|$$

A small Delta score means a higher degree of stylistic similarities. During the attribution, the document is assigned to the author with the lowest Delta value, which implies the greatest similarity between the document and the author's class. Initially this metric was used with 150 most frequent words to assess the dataset of Restoration poets and was considered highly effective.

Hoover (2004) introduced several improvements to Burrow's classical Delta, including ignoring personal pronouns, considering different numbers of frequent words (n = 700 was claimed to be optimal), and using culling at 70% (that is, ignoring the words for which a single text provides 70% occurrences or more).

Different approaches have been proposed to the calculation of the Delta score since then (Hoover, 2004; Hoover, 2005; Burrows, 2007; Argamon, 2008; Jannidis et al., 2015; Smith and Aldridge, 2017), including those using percentage difference instead of z-score and various metrics to calculate the difference between z-scores of a pair of texts. Along with classical Delta (which uses a variant of Manhattan distance), Cosine similarity and Canberra distance have been successfully used, notably in the Stylo package for stylometric analysis (Eder et al., 2016).

Z-score can be used separately outside the Burrow's Delta (Juola, 2006) to compare expected and observed frequencies of various words, not only the most frequent ones. This approach enables discovering the words over-used and under-used by each of the authors.

Instance-based similarity techniques were also used for compression models. Benedetto et al. (2002) used compression algorithms as means of calculating a pairwise similarity between texts in the way resembling one used in (Marton et al., 2005). The difference is that the target text is concatenated with a single training text, compressed, and compared with the compressed training text before concatenation. After that, a 1-nearest-neighbor classifier is used to attribute the target text to one of the authors.

Distance-based approaches focus on calculating the distance between the query document (Q) and the author class directly, without prior representation of both in a multidimensional space. The distribution of words can be treated as a probability distribution and represented using existing probability difference metrics, such as Kullback–Liebler divergence or Kolmogorov–Smirnoff distance. Cilibrasi and Vitanyi (2006) suggested an alternative distance metric based on Kolmogorov complexity. Kolmogorov complexity (Li and Vitanyi, 1997) of a given pair of strings defines the smallest program that converts that string into another one. For the task of AA, Kolmogorov complexity can serve as a similarity metric in which the text that requires the least effort for being converted into the target one is assigned the highest similarity.

Kukushkina et al. (2001) made use of Markov chains for classification. A first-order Markov model was calculated separately on each author's training set, and the target text was attributed to a certain class if its chain yielded the highest probability of producing this text

Vector space models

Training and target texts can be represented as multidimensional vectors in which each dimension corresponds with a certain feature. In this framework, a vast number of machine-learning approaches can be applied, including Support Vector Machines (SVMs) (de Vel et al., 2001; Li et al., 2006), decision trees (Zhao and Zobel, 2005), and neural networks (Zheng et al., 2006; Khosmood and Levinson, 2006).

SVM solutions turned out to be particularly efficient due to their ability to handle noisy or sparse data and process multidimensional vectors without overfitting. They learn the location of hyperplanes separating the data in the training set in the best way and with highest possible resistance to classification error. They have been extensively used for authorship attribution, outperforming such statistical techniques as Linear Discriminant Analysis (LDA) and many other machine learning solutions, such as Naive Bayes and Classification Trees.

In modern works, they are frequently used as one of the baselines, and are still showing compatible results (Stamatatos et al., 2018).

Deep learning models

Frequently used neural networks that are atypical for other NLP tasks are CNNs for comparing selected excerpts of n-grams between texts (Ordoñez et al., 2020) and Siamese architectures networks that were introduced specifically for detecting similar entities (Koch et al., 2015), though were initially used for comparing images.

A Siamese architecture generally incorporates a neural network that is applied separately to several (normally two) instances using the same weights in order to produce comparable representations. These representations are then passed to a distance metric to determine their similarity. For the training stage, the production of representations can be adjusted to maximize the distance for a particular task, and for the inference, a threshold can be used to classify a pair of data entries.

The specific model used within the Siamese architecture can be different.

One approach was proposed by Boenninghoff et al. (2019) for attributing short texts. The An AdHominem model used two layers of bidirectional LSTMs with an attention layer on top for both word and character embeddings, and a module for nonlinear metric learning was used to calculate the similarity between their outputs.

Tyo et al. (2021) generated embeddings of a fine-tuned BERT and then used average pooling across all tokens for each of the input sequences and a dense layer to obtain a final representation of a sequence. After that, cosine and Euclidean deltas as distance metrics. For the training the distance was used as means of loss calculation and model optimization, while for the inference, a threshold was used to classify the pairs.

Interestingly, Koppel et al. (2012) considered similarity-based approaches more appropriate than machine-learning methods for attribution among many candidate authors. However, it is not clear to which extent is this claim applicable to large modern ML models for text classification.

2.4 Authorship attribution datasets

Neal et al. (2017) provide a thorough review of the datasets available for authorship attribution and commonly used for such task. We, in turn, will outline the major steps of their development, highlighting the importance of cross-topic data representation, and proceed with discussing PAN datasets in some more details.

2.4.1. Traditional datasets

Traditionally, AA was performed for texts that possessed considerable social and cultural importance, such as biblical literature (Morton and McLeman, 1966; Kenny, 1981; Eder, 2012), philosophical works (for example those of Plato and Aristotle (Campbell (1867)), or major literary works (Mosteller and Wallace, 1964; Merriam, 1998).

The Federalist Papers, investigated by Mosteller and Wallace, were chosen as they satisfy a number of requirements that clarify and facilitate the attribution procedure: the texts are publicly accessible, the potential authors are known with certainty (which makes it a closed-set AA), and the training set is already defined since some of the 85 essays have been signed by the authors. Besides, these texts are homogeneous in terms of genre and theme and were published in the same sources during a limited time frame.

2.4.2 Modern cross-domain datasets

In automated AA, computational models enable handling larger datasets and provide predictions more efficiently even when the information about authors is limited.

The availability and structure of textual data play a key role in such tasks. In (Barlas and Stamatatos, 2021) numerous data-specific subfields are outlined, including AA in digital humanities (that of historical works) and in social media analytics (identifying the authors of tweets and other social media microtexts). A common problem with complex datasets is that the training subset (with known authors) and test one (with unanimous or questionable authorship) may have different properties. In some cases, we only possess training data of texts in other genres or on other topics. To address this issue, a substantial number of current research projects in AA make use of cross-domain attribution.

CMCC (2009)

Training a cross-domain model requires a highly elaborated dataset in which domain and/or genre parameter can be isolated. The earliest corpus of such type was introduced by Goldstein-Stewart et al. (2009) and is sometimes referred to as CMCC. This is a controlled corpus with respect to the genre, topic, and demographics of subjects. It contains excerpts of texts from 21 undergraduate students in six genres ("blog, email, essay, chat, discussion, and interview") and six topics ("church, gay marriage, privacy rights, legalization of marijuana, war in Iraq, gender discrimination") written in English. It enables the evaluation of AA methods in cross-topic and/or cross-genre settings, ensuring that other factors that can affect performance (e.g., demographics of authors, distribution of samples over the authors) are diminished.

It was created for the task of person identification in 2009. The authors of the original dataset applied four classifiers (Naïve-Bayes, SVM, decision trees, and random forests) and achieved 82% accuracy in some cross-genre and 94% in some cross-topic tasks.

The dataset has been widely used in AA since then. In Stamatatos (2017) the corpus was used in three settings (cross-topic, cross-genre, and cross-topic-and-genre) to assess the improvement in the performance of C3G-SVM and PPM5 models by introducing distortion techniques. In (Sapkota et al., 2014) the authors investigated the performance in single cross-topic and multiple cross-topic conditions using this corpus along with others, and the same was done in (Barlas and Stamatatos, 2021) for the assessment of transfer learning AA. In (Stamatatos and Barlas, 2020) the corpus was used to assess the performance of pre-trained models.

The Guardian Corpus

Another corpus used in (Sapkota et al., 2014) is called The Guardian Corpus. The corpus was constructed using the public API of The Guardian which enables search by keywords, authors, and topics. The original corpus was created by Stamatatos (2013) and included opinion articles on four topics as well as some book reviews.

An extended and balanced version of the Guardian dataset is provided by Altakrori et al. (2021). In this dataset, each author is associated with 40 documents.

PAN 2018

A cross-domain dataset was created in the framework of AA task at PAN 2018 (Stamatatos et al., 2018), an annual competition and workshop on digital forensics and stylometry. The dataset includes subsets for multiple AA problems in different languages, but in (Barlas and Stamatatos, 2021) the authors only consider English texts from this dataset.

The corpus incorporates fanfiction texts drawn from different fandoms, that is, dedicated to or based on various original works of art. Such distinct "universes" of storytelling are treated as topics that are consistent within themselves and differ from each other. In the PAN 2018 setting, the test texts with unknown authorship belong to one fandom, while the training texts of known origin are from numerous other fandoms. This makes the attribution a "cross-fandom" procedure (Stamatatos et al., 2018).

PAN 2020

PAN is an annual event, and the dataset has been improved over the last few years. The improved and extended version is introduced by Bischoff et al. (2020) and adapted for the PAN setting by Kestemont et al. (2020). Itis still based on fanfiction since it is considered among the fastest growing forms of literary texts accessible online. Fanfiction is openly available and easily scrapable from the Internet, which makes it a suitable candidate for a corpus of unseen literary works.

The PAN20 version consists of two datasets ("small" and "large"), both being significantly larger than the previous fan-fiction corpus. In the 2020 version, only English texts are included, and their average length is roughly 21000 characters. The "small" corpus contains 52601 texts, while the "large" one contains 275565.

2.4.3 Datasets discussion

Although numerous traditional datasets exist for the task of authorship attribution, Fabien et al. (2020) mention that a sufficient number of examples for each author is required. The texts themselves need to be sufficiently long: Koppel and Schler (2004) name the recommended length of 500 words for measuring style. Besides, the use of comparatively complex and large models is motivated for larger datasets with additional complications, such as the necessity of cross-domain attribution, for otherwise a simpler and better explainable classifier, such as one based on logistic regression on lexical features, would suffice.

Thus, it seems that the most appropriate choice for this research would be the PAN 2020 corpus, as it is the largest currently available dataset that has been specifically designed to assess the latest techniques of authorship verification. Another advantage of this choice is the potential ethical importance of the task, namely, detecting plagiarism and fraud to protect independent fanfiction writers.

3. Transformer models in authorship attribution

Transformers are a comparatively recent type of neural networks that were introduced by Vaswani et al. (2017). The title of their paper, «Attention is all you need», suggests that models based solely on attention mechanisms can be used without convolutional or recurrent architecture and achieve impressive results outperforming previous, more complex models.

Before the outbreak of transformer models, long short-term memory models (LSTMs) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRUs) (Graves, 2013; Chung et al., 2014) have been widely used for a variety of NLP tasks, where they showed a state-of-the-art performance (Cho et al., 2014; Sutskever, 2014)). However, their sequential nature — namely, that they relied on the iterative generation of hidden states as functions of previous hidden states and new inputs — limited their performance in processing longer sequences. Memory constraints "limited batching across examples" (Vaswani et al., 2017), and vanishing gradient problem (Bengio, Fraskoni, Schmidhuber, 2003) caused the decrease of the nodes' influence on the current state in the distance even when they had high weights. As a result, the dependencies between distant elements were hard to learn. Gated RNN architectures (Chung et al., 2014) achieved noticeable improvement in taking various parts of a long sequence into account, while factorization techniques for LSTMs (Kuchaiev and Ginsburg, 2017) increased their computational efficiency, but the underlying problem of sequential processing persisted.

Attention mechanisms were proposed as a method to address this problem (Bahdanau, Cho, Bengio, 2014; Kim et al., 2017) since they could learn dependencies between elements in the sequence regardless of the distance between them. However, until the emergence of Vaswani et al. (2017) paper these mechanisms were mainly used as part of the model on top of some RNN architecture (Paulus et al., 2017; Cheng et al., 2016). Attention was applied in a variety of tasks such as text summarization (Paulus et al., 2017) or machine reading (Cheng et al., 2016). The proposed Transformer model differed from them in that it only made use of attention to learn global dependencies between input and output.

An important advantage of this approach is that the maximal (worst-case) number of operations required to relate input from two positions is constant ($O(1)$) in the Transformer, while in RNNs it grows with distance ($O(N)$), where N is the number of elements between selected positions) albeit at a different rate with various tools for computational complexity reduction (Graves, 2013).

A valuable side benefit of Transformers is that the underlying self-attention mechanism could be used as a base for interpreting the models 'output. Inspecting and visualizing attention weight can reveal the role of each attention head in capturing semantic and syntactic features of a sequence and provide insights into how particular sequences are analyzed and which relations between tokens play the most important roles.

3.1 Transformer overview

Architecture

The original Transformer model also follows the encoder-decoder architecture (Cho et al., 2014; Sutskever et al., 2014; Bahdanau, Cho, Bengio, 2014) by using separate encoder and decoder modules, both based on self-attention. In the encoder-decoder framework, an input symbolic sequence is first mapped to a sequence of representations by the encoder, and then an output symbolic sequence is iteratively generated by the decoder given the sequence of representations. This model is auto-regressive (Graves, 2013) as it adds the output from the previous step to the input when generating the new output.

In Transformer, each layer of encoder consists of a self-attention (a type of attention that relates items in a single input sequence to create a representation of this input) and a fully-connected feedforward neural network. Decoder layers have two multi-head self-attention sublayers. The first one features a masked (restricted) attention that only computes attention weights between the element and its left context to ensure that prediction for the current output only depends on the previous output, i.e., the output is indeed generated in an auto-regressive manner. The second sub-layer uses regular self-attention but takes the encoder output as part of its input.

Later works experimented with Transformer models that only incorporate encoder- (Devlin et al., 2019) or decoder-type (Radford et al., 2018) attention modules to make use of their specificities.

Attention Mechanism

Attention is a function that maps a query and a set of key-value pairs to an output. All these components are represented as vectors. The compatibility function computed using the query and the key yields the weight assigned to the corresponding value. This weight corresponds to the relevance of this item to the query.

Several types of attention functions exist, and the most widely used ones are additive attention (Bahdanau, Cho, Bengio, 2014) and dot-product attention (Luong et al., 2015). The former uses a single hidden layer for compatibility function, while in the latter dot products of the query and each key are computed and converted to weights using a softmax function. Vaswani et al. (2017) use scaled dot-product attention, which differs from the regular dot-product function in that it uses a scaling factor of $\frac{1}{\sqrt{dk}}$, where dk is the dimension of query and key vectors.

The authors' motivation for this adjustment is that scaling the dot products makes the softmax output more diverse and facilitates learning, as large dot products would otherwise push the corresponding softmax values to the limits.

Even though the performance of two attention functions has similar theoretical complexity (Vaswani et al., 2017), the dot products can be optimized using efficient matrix multiplication, which makes them favorable for training large models. In this case, sets of vectors are represented as matrices of queries (Q), keys (K), and values (V). For the decoder module, the softmax input for tokens from the right context is set to −∞ to implement masking.

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

Another specificity of the Transformer model is the use of multi-head attention, in which Q, K, and V are linearly projected to corresponding dimensions, attention is calculated for each set of projections in parallel, and the results are concatenated before the final projection to the original output dimension. Multi-head attention is added to prevent averaging the values in cases when two vectors show high compatibility only in some representational subspaces, which could be the case for a single attention head. The number of heads is a hyper-parameter that was explored by Vaswani et al. (2017). It was shown that, while multi-head attention improves the model performance (with best results achieved using 8 heads), too many heads are less efficient as the size of each head's dimension d_k becomes too small. The authors also claimed that the importance of the large d_k dimension may suggest that query-key compatibility has complex nature and other compatibility functions should be explored.

Positional Encodings

Since the attention function computes weights between all tokens in the sequence simultaneously, positional information needs to be inserted additionally in order to let the model make inferences based on the order. This can be done by adding positional encodings (PE) (Luong et al., 2015) to the input embeddings. Different types of PE exist, but a convenient choice used by Vaswani et al. (2017) is sine and cosine functions that allow linear calculation of PE for each offset given the original value.

$$PE_{(pos,2i)} = \sin(pos/1000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/1000^{2i/d_{model}})$$

The original Transformer model was assessed on a machine translation task, in which it outperformed previous best models and established a new state-of-the-art BLEU score. After that, considerable success has been achieved with this architecture on various NLP benchmarks for such tasks as question answering (Rajpurkar et al., 2016), sentiment analysis (Socher et al., 2013), language understanding through inference (Williams et al., 2018) and aforementioned machine translation (Papineni et al., 2002).

3.2 Transformer models fine-tuning

A particularly useful trait of transformers is their effectiveness in transfer learning, that is, pre-training on one task and transferring the obtained knowledge to another task with fewer available training data and more limited supervision possibilities, such as authorship attribution (Barlas and Stamatatos, 2021). In NLP, language modeling has become a widely used base task (Radford et al., 2018; Radford et al., 2019; Devlin et al., 2019) due to the abounding language data and comparatively easy training process, although supervised tasks, such as machine translation, have also been exploited for pre-training (McCann et al., 2017).

Peters et al. (2018), Radford et al. (2018), and other researchers have shown that pre-training a language model for applying to more specific downstream tasks can be highly effective in NLP. The central component of pre-training is usually a unidirectional language model that provides general language representation, while the specific implementation may differ. Devlin et al. (2019) distinguish two types of pre-training. In a feature-based approach pre-trained representations are used as additional features in an overall task-specific architecture. This technique is implemented, for example, with word embeddings in the skip-gram model (Mikolov et al., 2013) and context-sensitive ELMo vectors (Peters et al., 2018) and with sentence embeddings in (Logeswaran and Lee, 2018). In the fine-tuning approach used in OpenAI GPT (Radford et al., 2018), all original pre-trained parameters are fine-tuned for the downstream task so that few parameters need to be trained from scratch.

3.2.1 BERT

BERT, or Bidirectional Encoder Representations from Transformers, is a language representation model introduced by Devlin et al. (2019). Its distinctive feature contrasting with OpenAI GPT is that it uses bidirectional representations in all layers, which means that each self-attention is unrestricted for both right and left contexts. In the original Transformer, only the encoder component makes use of bidirectional attention.

The advantage of this design is that BERT can be easily and relatively inexpensively fine-tuned with only one additional layer on top of the original model. This makes this model preferable for various NLP tasks, such as language inference and sentiment analysis. Besides, BERT outperformed previous Transformers and other task-specific models in machine translation, question answering, and numerous other tests.

BERT architecture

BERT differs from the vanilla fine-tuning approach in that it adds bidirectionality for the language model to make use of context from both directions. According to (Devlin et al., 2019), this extension optimizes sentence-level inference and results in significant improvement for token-level tasks, such as question answering.

To this end, BERT introduces a "masked language model" (MLM) that performs random masking of several input tokens from the input for training and learns to predict the missing tokens based on their context from both sides. The resulting bidirectional representations are deep, unlike those obtained in (Peters et al., 2018) by a "shallow concatenation of independently trained left-to-right and right-to-left LMs" (Devlin et al., 2019).

BERT is first pre-trained on unlabeled data, and then the learned parameters are initialized for fine-tuning on labeled data. Same pre-trained data can be used for various downstream tasks though fine-tuning layer should differ.

In pre-training, the model takes a span of unlabeled text as input. This can be a single sentence or a concatenation of, e.g., question and answer. The first token in the vector representation of input is a classification token CLS that aggregates sentence representation. Sentence pairs are separated with a SEP token. Additionally, a learned segment embedding is added to indicate the sentence to which a token belongs. In the result, input representation consists of a word, segment, and position embeddings.

After that, training is performed in two ways. In MLM, the predicted vectors for the masked tokens are taken by a softmax over the vocabulary to generate the missing word. In the next sentence prediction (NSP), the second sentence is replaced with a random one in 50% of cases. This type of training is required for learning relationships between to sentences and obtaining sentence representations for such tasks as question answering. BERT can use both sentence embeddings and other parameters for a downstream task.

BERT was pre-trained on English Wikipedia and other corpora and assessed on GLUE benchmark (Michael et al., 2018), SQuAD question answering dataset (Rajpurkar et al., 2016), and SWAG language inference dataset (Zellers et al., 2018), where it showed state-of-the-art results.

The downside of pre-training a bidirectional unlabeled language model is that it entirely relies on unmasked self-attention, so that the weights need to be calculated between all pairs of tokens in the sequence. This limits the maximal length of the input sequence: in the best-performing BERT model the input is limited to 512 tokens. Given that some tasks involve concatenated text pairs, this can be a serious limitation for tasks in which a large chunk of text is required to create a valid representation. This constraint is one of those alleviated in GPT-2 (Radford et al., 2019).

Liu et al. (2020) presented RoBERTa, an optimized version of BERT with higher robustness. Longformer, introduced by Beltagy et al. (2020), is a model based on RoBERTa, but specifically adapted to process long sequences.

3.2.2 Transformer Language Models evolution

Starting from BERT, GPT, XLNet (Yang et al., 2019), Transformer-XL (Dai et al., 2019), and MASS (Song et al., 2019), the trend for more general language models with an increasingly large number of parameters continued. Several generations of models have been created trained on extremely large datasets using different variants of the transformer architecture, with the number of parameters growing roughly by ten times every year. The latest implementations include GPT-2 (Radford et al., 2019), GPT-J (Komatsuzaki, 2021), MegatronLM (Shoeybi et al., 2019), Turing-NLG (Rosset, 2020), T5 (Raffel et al., 2020), and GPT-3 (Brown et al., 2020). Recently a Megatron-Turing NLG model developed by NVIDIA was revealed (Kharya and Alvi, 2021). With over 530B parameters it is the largest language model up to date. It established a new state-of-the-art in LAMBADA test (Paperno et al., 2016) as well as in other metrics.

At the same time, the growing cost and complexity of collecting the data and training the language model motivate increasing interest in reducing the model size while retaining the previously achieved scores. One example of such effort is DistilBERT (Sanh et al., 2019), a smaller version of BERT which is 60% faster.

3.3 Enhanced transformers

Apart from reducing the size of the model, other improvements can be done to reduce computational complexity and increase the attention span for more accurate account of high-level textual features.

3.3.1 Computational cost

Despite the significant success of Transformer models, the computational complexity and memory cost remained serious limitations. Vaswani et al. (2017) compared the computational complexity of Transformers to RNNs and noted that self-attention layers are «faster than recurrent layers when the sequence length N is smaller than the representation dimensionality d», for the layer-wise complexity of global (unrestricted) self-attention is $O(N^2 \cdot d)$ compared to $O(N \cdot d^2)$ for a recurrent layer. This was indeed the case for the original machine translation application since the number of parameters was set to d = 512, and the model was given sentence pairs as input with words as tokens.

However, in practice quadratic complexity $O(N^2)$ limits the context size due to time and memory demands. This can become a critical bottleneck in cases where long-term dependencies need to be captured, such as text summarization (Paulus et al., 2017; Liu et al., 2018), question answering (Rajpurkar et al., 2016), or authorship attribution (Barlas and Stamatatos, 2020; Fabien et al., 2020; Ordoñez et al., 2020; Futrzynski, 2021; Peng et al., 2021) where the whole text needs to be considered to discover meaningful patterns. The demand for longer sequences increases given the common representation of such tasks for fine-tuned models, in which two sequences (source and target or two items for comparison) are concatenated and separated with a SEP token. For effective processing, all concatenated sequences and special tokens must fit into the limit of N = 512, which is a serious limitation.

3.3.2 Improved architectures

Increased input length

Transformer-XL introduced by Day et al. (2019) tried to solve the problem of limited context at the cost of computational speed. This model achieved state-of-the-art results in language modeling as it could learn dependencies beyond the regular context scope, but further increased the computational cost.

Longformers (Beltagy et al., 2020) suggested a compromise between length and computational cost by providing a model that could accept long sequences as input but processed them with a limited context window. It is an extension of BERT trained on ca. 6.5 billion words that established new state-of-the-art in Wiki-Hot and Trivia-QA (Joshi et al., 2017). It makes use of sliding attention window to compute local self-attention instead of a global one, thus reducing the computational complexity from quadratic ($O(N^2)$) to linear ($O(N \cdot w)$), where w is the attention window size. According to Beltagy et al. (2020), global attention can also be used to incorporate long-distance relations into special tokens, such as a class token CLS.

This approach goes in line with the original suggestion by Vaswani et al. (2017) when the Transformer was introduced: namely, that “self-attention could be restricted to considering only a neighborhood of size r in the input sequence centered around the respective output position”. Another approach to restricted attention was put forward by Suhkbaatar et al. (2019), in which optimal attention span for each attention head was learned.

Optimized Complexity

Other researchers aimed at reducing the complexity by optimizing the computation while retaining the global character of self-attention. Child et al. (2019) used sparse factorizations of the attention matrix and achieved the reduction of complexity up to $O(N\sqrt{N})$, while Kitaev et al. (2020) proposed a Reformer, an improved Transformer model that used locality-sensitive hashing (LSH) to reduce the complexity to $O(N\log N)$. As Katharopoulos et al. (2020) point out, this technique imposes a limitation on the values of keys: namely, that they need to be identical to the queries.

Efforts have been also made to increase the speed of the Transformer inference through optimized memory consumption. To this end, weight pruning (Michel et al., 2019), weight factorization (Lan et al., 2020) and weight quantization (Zafir et al., 2019) have been used. Besides, Lample et al. (2019) explored a different type of attention with product keys that was claimed to increase the capacity of each attention layer. However, according to Katharopoulos et al. (2020), the overall complexity remained quadratic with respect to the sequence length.

A new promising approach was proposed by Katharopoulos et al. (2020). They introduce a Linear Transformer Model that is declared to have linear complexity ($O(N)$) without any limitations. They propose a change from existing softmax attention to a feature map-based dot product attention that utilizes the associativity property of matrix products. This results in reduced time and memory demands and incredibly high inference speed, up to thousands of times faster for image generation. As the authors state, this type of transformer with linear attention can be expressed as an RNN as it can perform autoregressive output generation recurrently with attention memory and normalizer memory as two layers of hidden states. To our knowledge, this type of Transformer model has not been tested for NLP tasks so far, therefore its performance on existing benchmarks in this domain is yet to be assessed.

3.4 Transformers application for text classification

Among the downstream tasks to which transformers can be applied, some require classification of the input text (Shaheen et al., 2021; Chi et al., 2020), including sentiment analysis and emotion classification. The task of authorship attribution also comes down to attributing the text into one of the authors 'class, although traditionally categorization is based on stylistic features, and the transformers 'capability of grasping such features requires additional account.

3.4.1 Transformers for authorship attribution

Several attempts have been made to perform AA using transfer learning based on a Transformer language model. Barlas and Stamatatos (2020) observed the potential for using pre-trained language models for cross-topic and cross-domain AA. They assessed BERT, GPT-2, ELMo, and ULM-FiT (Howard and Ruder, 2018) supplemented with a multi-headed classifier. Each classifier was trained for a binary author verification task for each author, and the text was attributed to the class with the highest score of a corresponding classifier. In the result, BERT showed the best performance beating RNNs.

BertAA

Fabien et al. (2020) take this work into account and broaden it with an exhaustive analysis of pre-trained language models for AA with their limitations. They apply the models to three widely known corpora: Enron Email corpus by Klimt and Yang (2004), Blog Authorship Attribution Corpus by Schler et al. (2004), and IMDb Authorship Attribution Corpus by Seroussi et al. (2014).

They proposed a BertAA model for the authorship attribution task, and to our knowledge, this is the first transformer-based ensemble designed specifically for AA. This model used the BERT language model since it had been shown to perform well in classification tasks (Sun et al., 2020) and is capable of extracting semantic and syntactic information useful for stylometric analysis. To incorporate high-level stylistic information, they combined the original model with stylistic and hybrid features.

The pre-trained BERT model is fine-tuned with a dense layer and a softmax activation to output class probabilities, as suggested in (Sun et al., 2020). Weights of both the dense layer and the

BERT itself are adjusted. The Stylometric classifier is based on “style” features selected in (Sari et al., 2018), namely the text length, words counts, average word length, number of short words, the proportion of digits and of capital letters, frequencies of letters and digits, hapax legomena, and punctuation frequencies. Additional “hybrid” features include 100 most frequent n-grams (for n = 2 and n = 3). Stylistic and hybrid features are fed into two separate logistic regressions that also yield class probabilities. A final layer of logistic regression is then used to make the choice based on the probability distributions from three classifiers.

The model achieved state-of-the-art performance on the Blog Authorship dataset (Schler et al., 2004) but was outperformed by a CNN classifier on an IMDb62 dataset (Seroussi et al., 2014) with more authors and fewer data per author. It was shown that the inclusion of stylometric and hybrid features slightly improves the F1 score but affects the accuracy. The authors conclude that sufficient training data for each author is a necessary prerequisite for the successful performance of BertAA, while this is rarely the case in real-life applications. Among the suggested extensions to their work are additional pre-training of BERT on the target domain before adding the dense layer and experiments with other LM architectures, such as RoBERTa (Liu et al., 2020).

Transfer Learning Approach

Barlas and Stamatatos continued their work in 2021 with additional experiments on AA in cross-genre and cross-fandom conditions (Barlas and Stamatatos, 2021). They adapted the multi-head classifier from Bagnall (2015) where it was combined with an RNN language model but experimented with various transformer-based language models.

Compared to (Barlas and Stamatatos, 2021), this work explored the proposed methods in more detail and considered a cross-fandom task based on PAN corpus (Kestemont et al., 2018; 2019). The set of models remained unchanged. For BERT and GPT-2, texts were split into chunks of 510 and 1024 tokens respectively.

The proposed architecture included a LM, a filter for LM representations and a multi-head classifier (MHC). The filter ensures that only N most frequent tokens will need to be predicted by the MHC: specifically, it filters out those representations that are not followed by a token from N most frequent ones, though information about them is incorporated in accepted representations.

BERT and ELMo achieved overall best results in cross-topic task, but ULMFiT (Howard and Ruder, 2018) has shown comparable results in cross-genre AA. For the cross-fandom task, all models showed averaged performance below the baseline, although in some runs ELMo and GPT-2 achieved the best results. The performance representations from different layers of BERT were compared since they can capture different types of linguistic information (Jawahar et al., 2019), and shallow layers were shown to be slightly more effective. The authors also postulate the importance of normalization corpus and the difficulty of combining character-level and token-level information, which may suggest that ensemble models could be desirable.

PAN transformer-based solutions

PAN dataset (Kestemont et al., 2018; 2020; 2021) has been used for an AA competition, and four Transformer-based models have been proposed so far for the authorship verification task. The first one was suggested by Ordoñez et al. (2020). Their solution relies on the Longformer architecture to better model long text chunks. They use a concatenation of 511 consecutive tokens from each pair of texts with a SEP token as input and an attention window of 512 tokens. The CLS token from the Longformer output is combined with the document topic information and fed into the classifier, which is a multilayer perceptron. The model achieved above baseline performance on the validation set (F1 score 0.96 for the large version of PAN 2020 dataset) but performed significantly worse during the competition, reaching only F1 0.75 on the test set.

During the next-year event, Tyo et al. (2021) proposed a Siamese network that made use of BERT embeddings. The model was trained to locate embeddings of similar texts adjacently in the representation space, and a threshold was used at the inference stage to make the prediction.

Transformer-based model by Futrzynski et al. (2021) used vanilla BERT architecture with an input size of 30 tokens. They considered both the prediction based on single input and on 100 input sequences, from which a final output is calculated using the median of each component across all 100 output vectors. This model achieved F 0.76 but could not beat the baseline solutions for the PAN competition (Kestemont et al, 2021).

Solution of Peng et al. (2021) utilized a similar approach with multiple input sequences but used a larger input size (256 tokens) and only 30 sequences. Instead of using the median value across all vectors, they trained an additional classifier for the BERT output, to which global average pooling was applied. The model achieved F1 0.945 on the validation set and 0.917 on the test set, thus being the best solution proposed at PAN 2021 among those trained on the small version of PAN 2020 dataset (Kestemont et al., 2020). With all other metrics considered, their solution ranked second.

Thus, it can be seen that two general strategies are used to incorporate BERT architecture into an authorship verification model. In the first approach, the task is treated as a recognition of relations between segments from different texts. In such a case, a separator token is used to divide the segments, and the CLS token is assumed to learn how to represent stylistic relations between two parts. In the second approach, we only assume that CLS is capable of representing a text in the embeddings space, including the representation of its style. Therefore, two such representations can be compared using a distance metric, such as a dot product or a cosine delta. and the model can be fine-tuned to minimize the distance between the texts of the same author or maximize that between texts of different authors.

4. Transformer explainability

4.1 Defining explainability

The problem of explainability can be subdivided into three notions: transparency, explainability, and interpretability (Wiegreffe and Pinter, 2019). These terms have been used in different senses (Lipton, 2016; Doshi-Velez and Kim, 2017; Rudin, 2018; Riedl, 2019). Lipton (2016) pointed out that transparency largely conflicts with explainability. Transparency in his view refers to human understanding, the way a certain component of a model corresponds to (or can be mapped to) an understandable human construct.

From this perspective, attention is transparent as it provides a matrix of scores for each element of the input, which shows this element is weighted in the following steps, and which contribution it has to the attention output.

Explainability, on the contrary, refers to the extent to which the model performance can be analyzed post-hoc. It can be regarded as mimicking human explanations of various actions as it aims at justifying the decision and providing arguments for it.

Rudin (2018) defines explanation as a reconstruction of decision-making process and emphasizes that it is not necessarily a faithful reconstruction. Despite potentially erroneous underlying methods, explanations are considered important and human (Riedl, 2019) and were shown to increase the user's trust in a system (Thorne et al., 2019), which is especially important in sensitive domains.

The dichotomy between explainability and interpretability, in turn, is related to the level of abstraction. According to Rudin (2018), interpretability involves a holistic understanding of the relations between input and output (in this sense, shallow classification trees are typically interpretable). In (Doshi-Velez and Kim, 2017) a more fine-grained categorization of interpretability is presented.

While the explanations as interpreted by Rudin aim at resembling some human rationale, there is no full agreement on how their resemblance should be measured. Hence techniques of evaluating explanations also differ even within the topic of transformer explainability. In Lei et al. (2016), where explanations are generated at the same time as predictions and come directly from the input, the model is trained with gold-label explanations. Other works (Mullenbach et al., 2018; Ehsan et al., 2019) rely on human evaluation.

A number of techniques have been utilized to provide explanations for NLP tasks. Ross et al. (2017) measured feature importance using gradient information, while (Li et al., 2016) based the explanation technique on "representation erasure" in which the impact of each dimension from the representation being removed is calculated. Ghaeini et al. (2018) used visualizations of LSTM gating signals as well as attention saliency. In (Alvarez-Melis and Tommi, 2017) a causal framework for explaining predictions was proposed To explicitly identify explanations of black-box predictions.

4.2 Attention weights as an explainability tool

Vaswani et al. (2017) claimed that investigating the self-attention mechanism can lead to interpretations of the models 'output. Visualized attention weights can potentially reveal the function of each attention head in capturing semantic and syntactic features. For any particular sequence, these weights could also provide insight into the way it is analyzed by the model, in particular, which relations between words are captured.

Models that have at least one attention layer provide a distribution of attention weights over input units that is often treated as communicating the relative importance heatmap (Jain and Wallace, 2019). This assumption suggests that high attention weight of a certain input increases tis responsibility for the output of the model. Xu et al. (2015) used attention scores as means of visualizing the saliency of image content for the task of image description. Li et al. (2016) claimed that «Attention provides an important way to explain the workings of neural models», and multiple authors presented works in the spirit of this view, including (Mullenbach et al., 2018; Ehsan et al., 2019; Choi et al., 2016; Martins and Astudillo, 2016; Xie et al., 2017; Thorne et al., 2019; Serrano and Smith, 2019). Attention weights was also shown to correlate with human attention in the task of document classification (Pappas and Popescu-Belis, 2016).

Another line of work involved modifying the attention to facilitate interpretability, for example, by employing sparse attention that only uses a subset of inputs for prediction and considering this subset responsible for the output (Lei et al., 2016; Peters et al., 2018) or by fitting the attention weights to explanations provided by humans. (Bao et al., 2018)

4.2.1 Critical view at attention explainability

However, the degree to which attention can serve as an explainability tool is often questioned, and the specificity of the relation between attention weight and outputs is not completely clear for some researchers.

Jain and Wallace (2019) pointed out that attention weights lack explainability potential since they often show a weak correlation with gradient-based feature importance measures, and multiple distributions of attention weights can yield similar predictions in classification tasks. They claimed that two properties need to hold for a valid explanation: (i) that attention weights correlate with more reliable measures and (ii) that “counterfactual” and “adversarial” attention weights that substantially differ from original ones result in correspondingly significant changes in the output predictions.

They selected gradient-based feature importance scores and leave-one-out (LOO) as reliable measures as they had been shown to reveal individual feature importance with known semantics (Ross et al., 2017), even though they cannot serve as a standalone interpretation technique (Feng et al., 2018) due to inherent complexity of reaching neural models interpretability. The “counterfactual” attention distributions were generated at first by random permutations, while the “adversarial” one was calculated as a maximally different distribution in which the output prediction remains unchanged.

They assessed a BiLSTM and a feedforward encoder, both with two types of attention mechanism, on a number of binary classification tasks, question answering, and natural language inference, and found that these properties were not fulfilled by most models except one for the MIMIC (Johnson et al., 2016) classification task (for the positive outcomes subclass).

Firstly, correlation analysis revealed that attention weights don’t agree with standard feature importance scores in a strong or consistent way, albeit correlations between attention weights and gradient- or LOO-scores were much stronger for feedforward encoder than for BiLSTM.

Secondly, “adversarial” attention indeed yielded essentially unchanged predictions quite frequently, which led the authors to the conclusion that equally plausible explanations were possible for the same output, and hence none of them are reveal causal relation (that a model made a certain prediction because it paid attention to inputs in a particular way).

These considerations persuaded the authors that using attention to explain the models’ prediction can be misleading as it doesn’t provide meaningful insights; therefore, no indication of why a model made the prediction can be made and no “faithful” explanation can be created. They attributed the lack of interpretability potential of attention to the fact that representations encode arbitrary interactions between inputs and using attention weights of such representations cannot explain the output from the perspective of the input.

Another possible explanation was that only a few (“top-k”) features out of those used by a model show strong agreement with the feature importance measures. However, separating them would be complicated.

Importantly, this claim was only made for RNN-based models with attention layer(s), and future investigation was required for attention-centered Transformer models. Also, they noted that not all NLP tasks equally crave explanations: specifically, that interpretability is more crucial for classification tasks than for translation. This consideration may motivate our attention to the explainability of authorship attribution models since AA essentially belongs to the domain of classification problems.

4.2.2 Justification of attention explainability

Wiegreffe and Pinter (2019) joined the discussion with an opposite opinion explicitly conveyed by the title, “Attention is not Explanation”. They asserted that the explanation potential of attention depends on the definition of explanation and additional experiments are required to properly assess it, involving more layers of the model.

They agreed with the conclusion that attention fails to meet the criterion of consistency (i) and that inconsistent correlation with other interpretability techniques questions the validity of such explanations. However, they proposed four new tests instead of permuted and “adversarial” attention for criterion (ii), namely: a baseline with uniform attention to test the attention's contribution to the model; an examination of expected variance using multiple random input seeds; a diagnostic tool that used attention distributions as frozen weights in a multi-layered perceptron; a model-consistent end-to-end training protocol for adversarial attention.

According to the authors, key assumptions of the original experiment leave too much freedom in the setup for two main reasons. First of all, Jain and Wallace (2019) detached attention output prediction from the parameters used to compute them, and treat each attention score as independent of the model, which disregards the model itself. The explainability of attention weights is based on the fact that the model was trained to yield such weights.

The computation of “adversarial” distribution was also performed independently for different attention weights. Besides, “Existence does not Entail Exclusivity” (Wiegreffe and Pinter, 2019), which implies that providing an explanation instead of a comprehensive list of all valid explanations, as demanded by Jain and Wallace, does not make the result “unfaithful”. The final layer of LSTM produces complex outputs that can be aggregated and projected in different ways, and in case of a binary classification task, the demand for a single (or complete) explanation becomes even less justified as the attention matrix is reduced to a scalar attributed to one of two classes, and many dimensions corresponding to attention weights of certain inputs are not significant for this particular classification. This claim corresponds with Jain and Wallace’s hypothesis that only top-k attention weights are important.

Testing techniques

Wiegreffe and Pinter tested the validity of using attention for classification by comparing it to a simple baseline with uniform distribution of attention and discovered that in some datasets they do not perform better. Then they assessed the variance of attention weights by multiple training sequences.

For the third test, they used pre-trained weights from the attention-enabled LSTM model in a different model without recurrence (specifically, a multi-layered perceptron, MLP). The pre-trained attention scores performed well compared to MLPs with weights being uniformly distributed or learned from scratch, which suggested that attention could bear some model-independent interpretation of input tokens for a particular task on which they were trained.

Finally, they proposed a model-consistent protocol for training «adversarial» attention. This technique utilized a modified loss function that considered the distance from original attention scores. In cases where «adversarial» attention could lead to similar prediction output, the produced weights differed in a less extreme way compared to the original solution (Jain and Wallace, 2019). Attention scores obtained using this protocol did not perform as good as MLP weights, showing that attention distributions with the same output may not be equally powerful as explanations of the model, and it is the training on a specific task that infuses them with explainability potential. Thus, Jain and Wallace's claim that two explanations with the same output are equally possible doesn't seem so persuasive in light of this finding.

All these tests can in principle be used for assessing the explainability of a particular attention-based model.

Attention and Explainability definitions

Another contribution of the Wiegreffe and Pinter's work is a reflection on the nature of explainability, showing that the human-centered understanding of explanations does not require a unique and faithful rationale, and the plausibility of potential explanations should be assessed by human experts. They warn against confusing the demand for explainability with that of transparency (Lipton, 2016) and interpretability (Rudin, 2018), which are barely achievable for deep learning models due to their vast and complex structure. Demand for a single comprehensible explanation of which attention weights pattern leads to a certain output is in fact a demand for transparency, which is valid but likely unfulfillable.

4.2.3 Attention explanations in NLP transformers

The aforementioned techniques for explaining the models' performance based on the attention weights were tested for architectures that involve some layers of attention while based on LSTM or RNN architecture. Meanwhile, the Transformer models are centered around multi-layered attention, and specific explainability techniques may be used for them.

Vig and Belnikov (2019) analyzed the attention structure of a Transformer language model, namely a small GPT-2, and concluded that different features are grasped by attention on different layers. In order to consider the pattern of certain syntactic or semantic relations in the input, particular layers of attention need to be used.

Vashishth et al. (2019) attempted to give a more systematic account of information conveyed by attention layers in different NLP tasks to find out in which cases attention weights can be considered explanations (in support of Wiegreffe and Pinter (2019)) and in which cases they cannot (in line with Jain and Wallace (2019)). They differentiated between single-sequence (such as classification) in which attention «mimics gating units» and «pair-sequence» tasks (such as NLI and translation) and claimed that attention weights only have sufficient explanatory power for the latter. However, this paper aroused certain criticism as this distinction did not fully correspond with other findings in the domain.

A detailed account of the variety of functions performed by BERT attention heads is given in (Jawahar et al., 2019). They prove the presence of syntax-aware attention by finding specific heads attending to words with particular syntactic roles, such as direct objects of verbs, objects of prepositions, determiners of nouns, and use probing classifiers that take attention maps as input to evaluate their accuracy, which in some heads exceeds 75%.

They extract attention maps produced by BERT for 1000 random excerpts from Wikipedia articles. First, general surface-level patterns are examined, such as next- or same-token attention. After that, the potential of different heads to predict the word in a certain syntactic role is observed.

4.2.4 Transformer explanations beyond attention

Most methods overviewed above employ attention scores (i.e. products of queries and keys) ignoring other components of attention (queries, keys, and values separately) and other parts of the model. Hence, novel methods for transformer explainability can make use of these additional components.

Two such methods, attention flow and rollout method, were suggested by Abnar and Zuidema. (2020). The former involved the max-flow along the pair-wise attention graph and showed a considerably high correlation with gradient-based methods, but was computationally expensive. The latter made a linearity assumption for the layer-wise attention combinations, which led to erroneous emphasis on some tokens. However, it was adopted by other researchers (Dosovitskiy et al., 2020).

Another method was recently proposed by Chefer et al. (2021) who used Deep Taylor Decomposition principle to obtain relevancy scores. Their solution was designed for transformer-based computer vision tasks but was also tested for text classifications.

5. Model for authorship verification

In this chapter, we briefly introduce the problem that shaped the authorship verification task that our model is designed to solve and formalize the task. We then provide a more detailed analysis of the fan-fiction dataset that is being used, proceed with motivating the choice of a particular architecture and model for fine-tuning and introduce the baselines. After that, a detailed account of our implementation of the BERT segment classification model is provided with emphasis on the model hyperparameters that were investigated and altered compared to the originally suggested architecture. Finally, we introduce the results for different input representations and model sizes, compare them with existing implementations and reflect on the potential trade-offs, including the feasibility of using a larger model in practical applications. Finally, we outline potential paths for further research.

5.1 Model task overview

We begin our description of the attributional problem addressed in this research, the corresponding dataset, and the model we developed to tackle it, with a brief overview of the motivation behind it.

The rationale is described in detail in Kestemont et al. (2021). The primary goal of the recent PAN authorship identification tracks is declared as the struggle to verify the hypothesis of individual authors' stylome, as formulated by van Halteren (van Halteren, 2005). According to it, each author — or, in a wider sense, each writing individual — has an individual stylistic footprint that can be retrieved when a sufficient sample of one's writing is present. Despite a plethora of successful experiments in authorship attribution, these results still cannot be generalized to verify this hypothesis (Kestemont et al., 2021), with one of the major obstacles being the ad-hoc character of these experiments: a particular model can reliably discriminate between authors in a small group and tailor to their writing idiosyncrasies. Widening the model's success to an open-set case, especially that of hundreds or thousands of authors, however, is supposed to be more challenging, as it requires learning more general traits that separate individual styles without overfitting on particular authors' data.

These considerations motivated the creation of the largest to-date corpus specifically designed for the authorship identification tasks in order to challenge the generalization capacity of existing architecture and enable the development of more complex deep learning models. Given the size of this dataset and the fact that it has been specifically designed to assess the latest techniques of authorship identification, we believe it to be the most suitable for training the Transformer-based model. Another advantage of this choice is the potential ethical importance of the task, namely, detecting plagiarism and fraud to protect independent fan-fiction writers.

Given the limited number of texts available for each author, the demand for a large dataset motivated the involvement of a substantial number of authors. In such setting, solving the authorship attribution problem straight away may be perplexing, because locating the text accurately in the multi-dimensional space of several thousand authors is a non-trivial task. This becomes increasingly difficult in the open-set problem where the test set may include texts written by previously unrepresented authors. In practice, many attributional experiments are indeed performed in the open-set format, since the attributional hypothesis is typically based on extra-linguistic evidence and is rarely comprehensive. It is often re-formulated based on the results of the attribution procedure: if the target group of texts didn't show sufficient proximity to any of the a priori author classes, it can be concluded that these texts belong to an unrepresented class, and new candidate authors may be added. Differentiating between sufficient and insufficient proximity is complicated in a large space of authors, such as the one in our problem.

Therefore, the last three PAN tracks were concentrated on authorship verification, and the dataset was tailored to that problem. Authorship verification is essentially a binary classification problem, in which the goal is to detect, for each given pair of texts, whether they are written by the same author or not. Formally, it can be formulated as an approximation of the target function φ: (Dk, Du) $\rightarrow$ $\{T, F\}$, where Dk denotes a set of documents, all created by the same author, and Du represents a document of unknown origin. The function takes the value T if the latter also belongs to this set of texts written by one author, and F otherwise. For the task in question, the case of $|Dk|$ = 1 was considered, that is, the function only accepted a pair of texts.

Thus, for the current task, the goal of the model is to correctly detect whether each pair of excerpts from fan-fiction literary texts is written by the same author or not.

5.2 The PAN fanfiction dataset

The PAN 2018 and PAN 2020 datasets are overviewed in Section 2.4. Here, we will provide a more in-depth analysis of the "small" variant of the PAN 2020 set.

Total # of Texts	Same/Different Author	Number of Fandoms	Max length, Characters	Min length, Characters	Avg length, Characters
52601	27834/24767	1600	296887	20670	21425

Table 5.1: *The details of PAN 2020 authorship verification dataset*

A summary of some basic properties of this dataset is provided in Table 5.1. The dataset includes 52601 texts, most of them being close to the average length in characters (around 21500), but some featuring a significantly bigger length. The number of Same Author pairs is slightly larger than that of Different Author pairs and constitutes approximately 52.9%, which can be taken as a trivial majority classifier baseline accuracy.

Despite having a much smaller sample of authors, the dataset has the same number of fandoms, or topics, as the "large" counterpart – that is, 1600. The distribution of texts across these fandoms is not equal: a rather small share of fandoms is substantially more popular, followed by a large tail of fandoms with around 50 texts per fandom. The distribution is presented in Figure 5.1.

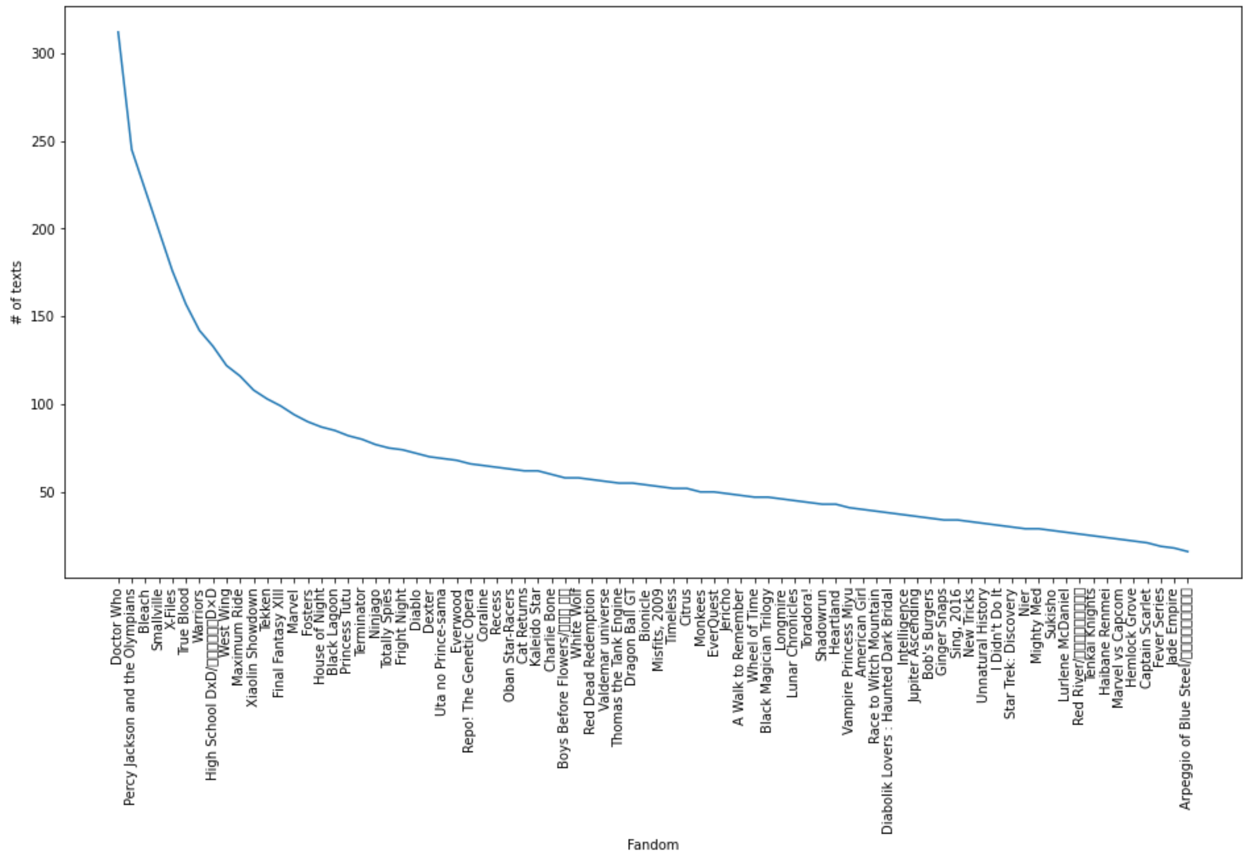


Figure 5.1: *Distribution of the fandoms. Every 20th fandom name is plotted on the X axis to exemplify the relative popularity of different fandoms.*

The exact number of texts for the 20 most productive fandoms is presented in Table 5.2.

Doctor Who	312
Avengers	280
Twilight	275
Criminal Minds	270
Supernatural	267
X-Men: The Movie	266
Pirates of the Caribbean	266
Harry Potter	265
Once Upon a Time	265
Vampire Diaries	264
Hetalia - Axis Powers	262
Lord of the Rings	262
Power Rangers	262

Legend of Zelda	259
Fruits Basket	257
Pokémon	254
Sonic the Hedgehog	253
Sherlock	250
Star Wars	247
Yu-Gi-Oh	245

Table 5.2 *Names of the most frequently occurring fandoms in the "small" version of PAN 2020 dataset*

The dataset consists of two files, with the first one containing the pair of texts and the information about the fandoms they are related to, and the second one providing the author IDs and the label, which has a binary value. The label takes the "True" value if both texts are written by the same author and the "False" value if the authors are different. The goal of the model is therefore not to identify particular authors of a given pair, but to predict that label.

In order to ensure that the dataset can be split into training and test subsets properly, we had to check that there is no excessive number of recurring instances. It was observed that, indeed, there are some texts that appear twice or more times, but these are only texts from the same-author pair. This re-occurrence can be explained by the procedure used to create the same-author subset of the corpus.

The same-author pairs were created by building all possible combinations of 2 out of n $\binom{n}{2}$ – that is, all pairings of texts within the author's subset, – without allowing the two texts to belong to the same fandom. However, the "small" dataset only contains a subset of pairs, and we typically observe 2 to 4 occurrences of each text, while many texts from same-author pairs still only appear once.

Besides, in many cases, abstracts are taken from different parts of the same text. They may overlap, but the beginnings and endings of such texts are different. This is done whenever the source texts are longer than 21,000 characters.

For different-author pairs, no repetitions that start from the beginning of the text were observed. It makes sense given a much large space from which different-author pairs can be drawn.

During the model design phase, the dataset was shuffled and split into the test set of size 5261 (10%), the validation set of size 4734 (10% of the remaining data), and the training set of size 42606. The validation set was used for tuning hyperparameters and comparing specific architectures. When the model design was complete, the model was trained and evaluated once again on a newly shuffled dataset with the test set of size 7601 and the training set of size 45000.

5.3 The model implementation

5.3.1 Recent approaches to Transformer-based AA

Transformer models, particularly those pre-trained for language modeling tasks, were shown to perform well on multiple NLP tasks with some basic fine-tuning or with barely any additional training at all, as is the case with GPT-2 (Radford et al., 2019).

Their ability to capture long-distance dependencies and to learn syntactic relationships suggests that they can be capable of learning such a complex notion as individual style as long as it can be formalized in terms of particular lexical choices, punctuation specificities, syntactic preferences, and frequent patterns. They may therefore be able to use this ability to differentiate those individual styles in the task of authorship verification.

Indeed, Transformers have already been used for authorship attribution a few times with different results. While BertAA (Fabien et al., 2020) showed state-of-the-art performance on some datasets, it was outperformed by a CNN baseline on others. Their experiments also showed that combining BERT features with supplementary quantitative features is not necessary as it causes a rather moderate improvement in performance. Comparable results were achieved by a standard fine-tuned BERT model (Devlin et al., 2019) that performed well in the cross-topic tasks but failed to overcome the baseline in the cross-fandom application.

Additionally, three Transformer models were applied to the PAN 2020 dataset with different degrees of success (Kestemont et al., 2020; 2021). One of them was a Siamese network that compared BERT embeddings of both texts from the pair (Tyo et al., 2021). Another model utilized the Longformer model in order to analyze longer excerpts from both texts, each including 511 tokens (Ordoñez et al., 2020). In these two cases, however, the models failed to outperform the baselines on the test dataset, which is particularly striking for the Longformer model that achieved accuracy well over 90% on the validation set (the one available to the developers, as opposed to the test dataset used during the competition to evaluate and rank the models). Nonetheless, the third model, which made use of an ensemble of BERT models and then classified their averaged prediction (Peng et al., 2021), achieved consistently high results on both sets during PAN 2021 track: it was ranked 3rd according to the F1 score and 6th according to the overall score, which also penalized low-confidence predictions. Moreover, it achieved the highest F1 score among the models trained on the "small" dataset.

Thus, BERT turns out to be among the most commonly used Transformer-based language models that are being fine-tuned to perform authorship attribution. Choosing it as the core of our architecture seems justified by the impressive performance of some existing models that are based on it, and in our experiments, we follow the most successful to-date architecture for BERT-based authorship verification, that is, the one suggested by Peng et al. (2021).

Baseline selection

Several baseline classifiers have been released specifically for the PAN competitions.

The one provided for AA task in PAN 2018 is based on character n-gram features and uses SVM with a linear kernel. For PAN 2019, a slightly adjusted version of the SVM classifier with character trigram input was used.

A "naïve" baseline classifier (Kestemont et al., 2016) makes use of cosine similarities between character tetragram text representations normalized by TF-IDF. It bears conceptual resemblance to the Stylo (Eder et al., 2016) classifier module, with the exception that it uses larger n-grams and is run using the *sklearn* library on Python, rather than R. The "naïve" classifier can be used both for attribution and verification, but it gains significantly higher scores in the second task of PAN (F1 ~0.79 vs ~0.55) (Kestemont et al., 2020).

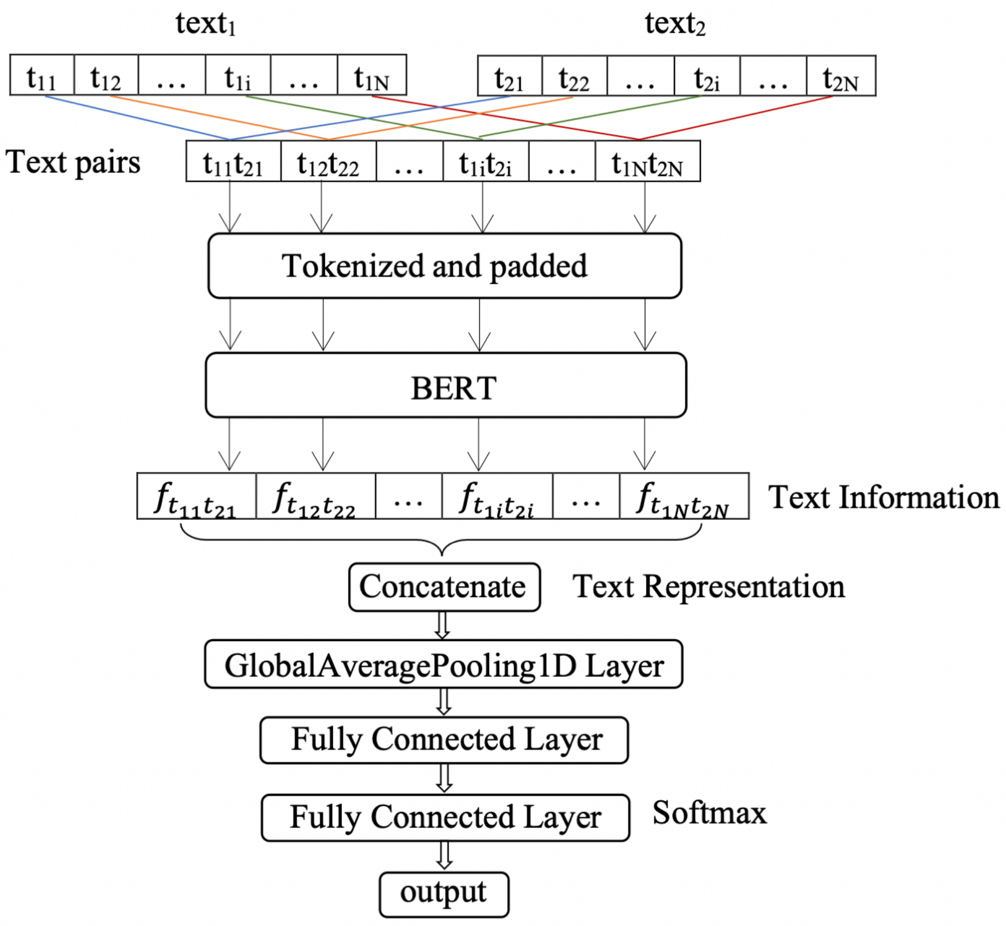
A "compression" baseline calculates the cross-entropy of a second text in the pair using the prediction by a partial matching model of the first text in the pair and then uses the values as input for the logistic regression classifier (Halvani and Graner, 2018). This baseline is based on the third-best model suggested at PAN 2018 and can also be used for the verification task.

Stylo (Eder et al., 2016) has been used as a baseline for several AA works, but its interface is better suited for attributing texts in a smaller space of candidate authors. Therefore, replicating the same authorship verification experiment using Stylo may require substantial processing of the data and classifying subsets of the dataset separately.

For the current experiment, we used the "naïve" classifier as the baseline and generated the explanations for that baseline as well.

5.3.2 BERT segment classification model

For this task, we adopted the architecture suggested by Peng et al. (2021). We provide a generalized description of that architecture, in which various hyperparameters and tools can be utilized to achieve results depending on the problem being tackled. The schema of that model is provided in Figure 5.2.



$text_1$
$text_2$
t_{11} t_{12} … t_{1i} … t_{1N}
t_{21} t_{22} … t_{2i} … t_{2N}
Text pairs
$t_{11}t_{21}$ $t_{12}t_{22}$ … $t_{1i}t_{2i}$ … $t_{1N}t_{2N}$
Tokenized and padded
BERT
$f_{t_{11}t_{21}}$ $f_{t_{12}t_{22}}$ … $f_{t_{1i}t_{2i}}$ … $f_{t_{1N}t_{2N}}$
Text Information
Concatenate
Text Representation
GlobalAveragePooling1D Layer
Fully Connected Layer
Fully Connected Layer
Softmax
output

Figure 5.2: *Architecture diagram for the model with original classifier module (from Peng et al. (2019))*

It revolves around splitting the two texts from the pair into segments and creating a new set of S pairs by combining them. For i-th segment from text 1, a corresponding i-th segment from text 2 is juxtaposed. If one or both texts were split into more than S parts, the remaining segments were left aside, so that the number of segments per text is constant in the dataset.

After that, the segments are tokenized and padded to have equal lengths across each segment in each pair, and the BERT model is used to produce embedding representations. Embeddings of CLS tokens, which are assumed to represent the information about the pair of segments, are extracted from the BERT output for each pair of segments and stacked in a two-dimensional tensor of size (S, 768), where S is the number of segments per text. The total size of the model's output for a given dataset will be (N, S, 768), where N is the number of instances in the dataset.

Finally, the tensor is averaged along dimension 0 to obtain a tensor of shape (N, 768). A simple FFNN classifier with two hidden layers of size 16 and 2 respectively connected with ReLU activation function is then used, and, after passing the output through the softmax activation, the model produces the probabilities for each class.

Such approach enables bypassing the central limitation of architectures based on BERT, namely the limited input length that can be at most 512 tokens – that is, 256 tokens for each text in the pair, including separation and class tokens. The use of multiple segments allows the model to still cover a large excerpt of the text while not exceeding the maximal input length.

We also point out additional benefits of the use of segments that can prove useful in authorship attribution. Firstly, the proportion of segments for each class prediction can be used as an indicator of the model's confidence, and a more reliable one than the class probabilities, for when the logits take extremely high and low values, their changes are not reflected in the final prediction. For particular practical applications, a threshold can be established to filter out unreliable predictions.

Secondly, the outputs of per-segment attribution can be utilized for dynamic analysis of co-authorship, in which we observe the parts of the text that are more likely to be written by a particular author out of a set of co-authors. Such type of analysis is implemented in existing AA solutions, such as Stylo package, and is frequently used for analyzing long co-authored literary texts.

5.3.3 BERT segment classification implementation

For the BERT model, Peng et al. used a package named *bert4keras* that utilizes Tensorflow backend. For the final classifier, a Keras sequential model was used. The potential downside of that approach is in the structure of dependencies that may lead to a versions conflict: to date, the last version of Tensorflow that works with that implementation (1.15) is unsupported by many versions of CUDA on local devices, as well as by multiple packages for explaining the predictions. Besides, both bert4keras package and the implementation of the model by Peng et al. lack English documentation and comments, which further complicates replication of their experiment.

Therefore, it was decided to implement the model with similar architecture from scratch, adopting only the snippet for splitting texts from the bert4keras snippets module. Our solution benefits from being implemented with a widely adopted HuggingFace API (Wolf et al., 2020), including data preprocessing with HuggingFace Datasets package, loading the tokenizer and the pre-trained BERT model with a sequence classification head from the HuggingFace Transformers library, and using the Trainer tool in combination with PyTorch backend to train the model. The final classifier was also implemented using PyTorch in order to reduce the number of dependencies.

Tokenizer

The input is being tokenized with the standard pre-trained *BertTokenizer* for the "*bert-base-cased*" model – that is, the regular-sized version of BERT that takes the casing of the input into account.

In general, BERT tokenizer in the Transformers library is based on a WordPiece algorithm for subword tokenization that had been proposed by Schuster and Nakajima (2012). Such classifier is initialized with all characters from the training data and learns the merging rules based on the probability of the combined sequence in the training data being greater than the probability of each subsequence combined. In the case of *BertTokenizer*, the tokenizer is already pre-trained like the BERT itself, using the same dataset. We are not performing any additional training since the tokenizer was exposed to a giant corpus of English language during the initial procedure and is, therefore, a reliable tool for splitting English literary texts.

Since the "*bert-base-cased*" model considers the casing, the corresponding tokenizer also treats differently cased words in a dissimilar way, most commonly assigning a word-level token to a word with conventional casing (the first letter capitalized in proper nouns, while for other words tokens are assigned to both capitalized and not-capitalized versions). Non-standard way of writing (such as fully capitalized words) is handled by subword tokens, which can be character-level n-grams or some affixes.

Of course, the vocabulary of subword tokens is not limited to those accounting for capitalization. There are no strict patterns, since the vocabulary was learned to optimize the performance: while some heavily affixed words are encoded with a single token, others are processed into morphemes or quasi-morphemes. The tokenizer does not necessarily follow grammatical rules of word formation: for example, the word "pleasingly" is tokenized into "plea", "sing" and "ly".

Other words that are typically decomposed into subword tokens are fandom-specific or otherwise unusual names that were probably previously unseen by the tokenizer ("R" + "ino" + "a"). Same stands for all typographic mistakes: "lengthy" is processed as a concatenation of tokens "le" + "ng" + "ht" + "y". Subword tokens are marked by a special "##" sign that allows the model to treat sequences of subword units in a specific way and avoid imposing excessive attention on their word-level combination.

BertTokenizer, as a variety of *PreTrainedTokenizer*, encodes input, token-type, and attention mask IDs simultaneously and is capable of detecting cases when the input is a pair of sequences. If this is the case, the input is encoded with an additional SEP token that separates the two sequences: [CLS] Sequence A [SEP] Sequence B [SEP]. Besides, token-type IDs are automatically added to indicate the sequence to which each token belongs.

Preprocessing

In our model, the tokenizer was applied to the first S segments of both texts, with S = 30. Segmentation was performed with help of the bert4keras snippet. It takes a list of separators and a maximal length value and concatenates the split sequences until they are about to exceed the maximal length. If a single sequence is longer than the maximal length, it is kept as is without truncation. After segmentation, the resulting number of segments in each text of the pair is compared with S. If larger, the remaining chunk is not passed through the tokenizer. If smaller, the list of segments is replicated 30 times until the requirement is met with certainty, and the first 30 segments from the resulting list are directed to the tokenizer.

The value was initially set to S = 30 following the experimental setup of Peng et al., but a more detailed investigation justified that choice.

We performed segmentation with two maximal length thresholds: 510 characters (as in Peng et al., ensuring that the majority of segment pairs reliably fall within the limit of 255 tokens after tokenization) and 750 characters (for the 512 tokens limit after tokenization), and the number of texts that had insufficient segments grew substantially when S was > 30. For segments with a maximal length of 750 characters, the number of pairs in which one or both texts didn't have 30 segments was 207 out of 52601. When S was increased to 31, the number of texts that failed to meet this requirement was 20366.

Such a striking difference is motivated by the fact that most texts have a very similar length of around 21000 characters. Therefore, S = 30 was shown to be a perfect compromise. It is the largest value that still enabled the vast majority of texts to be segmented without further augmentation needed, but still large enough so that the texts with many segments are not severely underrepresented after preprocessing.

Interestingly, the thresholds for the maximal segment length in characters are also rather tight. We have selected 750 as a compromise number: for segments of length 510, 64 pairs required augmentation, and after increasing the threshold to 750 the value grew to 207 pairs, which was still not a large percentage of data overall. However, after increasing the maximal length by mere 10 characters, over 2100 texts would require augmentation, and be the number increased to 800, the value would exceed 48000 texts.

Within these limits, nonetheless, we could adjust the preprocessing to ensure a more uniform length of segments. In the original implementation, Peng et al. used a limited set of separators (full stops, question and exclamation marks). This resulted in the separator being unable to split many texts into the required number of segments. After studying these examples, it was discovered that some authors favored separating sentences with symbols other than full stops and exclamation marks and used semicolons. To properly account for that idiosyncrasy and include such sentences without truncation, we added semicolons to the list of separators. That enabled us to reduce that number of augmented pairs from 64 to 59 in case of 510-character sequences and from 207 to 195 in case of 750-character ones.

When passing the segments to the tokenizer, the maximal length of the resulting encoded input was originally set to 255 tokens and then increased to 512 tokens. Segments that exceeded this length were truncated, and those shorter than the limit were padded with PAD tokens.

Each pair of segments was assigned the same label as the text from which it was excerpted. Datasets were converted to Huggingface Datasets format with the order of segments within one text preserved.

Fine-tuning

The pre-trained model used in our implementation is "*bert-base-cased*" from *AutoModelForSequenceClassification* with the number of classes = 2, which is essentially a BERT transformer model with a head for sequence classification. A head, in Transformer API terminology, is an additional wrapper on top of the base model class that adds extra output layers on top of the Transformer output (Wolf et al., 2020), which is a layer of raw hidden states. The layers used for the initial pre-training of BERT – that is, masked language modeling and next sentence prediction (Devlin et al., 2019), are also implemented as heads.

This model matches the first component of the proposed architecture as it enables fine-tuning BERT embeddings for our specific classification problem. We are using a standard $\text{BERT}_{\text{base}}$ configuration (L=12, H=768, A=12) where L denotes the number of hidden layers in the Transformer encoder, H represents the dimensionality of hidden layers, and A stands for the number of attention heads. The model is also initialized with an absolute type of position embeddings and the dropout probability for the hidden layer = 0.1.

Standard input model

For the fine-tuning stage, data was supplied in shuffled batches with *batch_size* = 30. PyTorch implementation of AdamW was used as the optimizer. AdamW is an Adam algorithm with weight decay (Loshchilov and Hutter, 2017). Initially, training was performed with the default learning rate of 5e-5, but the model performance did not increase over epochs, and the learning rate was decreased to 2e-5 in order to facilitate the learning process. The training was performed both using the Trainer API from the Huggingface module and using the native PyTorch training loop and achieved identical results.

In all training sessions, we used 5 epochs and evaluated the model's performance after each epoch to ensure that further training is justified. We have chosen that number following the Peng et al. (2019 approach and the recommendation of Sun et al. (2020) to keep the number of training epochs small. We observed that the best results, indeed, were achieved after training for 5 epochs. Moreover, additional training for 2 more epochs did not bring a significant improvement in accuracy and increased the model's eval loss, which suggests that the model may be overfitted with further training.

Training with a total number of instances being $N \cdot S$ = 1.35M typically took 27 hours on a local CUDA-enabled device using NVIDIA RTX3080 with 16GB VRAM.

Extended input model

Along with assessing the general capability of the architecture in question, investigation of the importance of the segment-wise input size on the model's overall performance. We exploited the longer segments with a maximal length of 750 characters and used a tokenizer with an extended input length limit of 512 tokens. The data was supplied in smaller chunks of 15 instances to comply with the increased memory requirement while preserving the integrity of segmented texts so that exactly two batches incorporate the segments from one text. The data was otherwise processed similarly to that in the base model, and the model was trained in the same fashion with identical hyperparameters. Training with the same resources took roughly 70 hours.

The fine-tuned models were evaluated using accuracy and F1 scores, both in the built-in Trainer evaluator function and native PyTorch loop. The results were identical.

In Table 5.3, we provide the outcomes for both standard and extended input models, including those after epoch 5, that were later used to produce segment embeddings, and results after epoch 4.

Model	Binary accuracy	F1
Untuned BERT	0.566	0.551
Standard input (epoch 5)	**0.852**	**0.847**
Standard input (epoch 4)	0.840	0.843
Extended input (epoch 5)	**0.876**	**0.872**
Extended input (epoch 4)	0.872	0.869

Table 5.3: *Performance of the model before and after fine-tuning for separate segments (before averaging)*

It can be seen that the model without fine-tuning is not capable of performing substantially above the chance level (which is 0.529). Both models perform better on the test set after 5 epochs compared to 4. The model with extended input, however, shows a noticeable advantage of roughly 2.5% over that with standard input size.

5.3.4 Obtaining embeddings

When the fine-tuning was complete, the model was used to produce segment embeddings for both train and test datasets. Two types of representations were obtained: firstly, we saved the CLS token embeddings from the last, 12th hidden layer of BERT. Secondly, we stacked and summed the embeddings of CLS tokens from the last four hidden layers, following the findings of Devlin et al. (2019) that showed that such representation may perform better in some tasks. Lastly, we obtained the logits from the classifier layer. All types of embeddings were converted in two shapes: a squeezed set of features with size (N, S, H) with N = 7601 for test and 45000 for train dataset, S = 30, and H = 768, and a flattened set with size (N * S, H) where N * S resulted in 228080 instances for test and 1350000 instances for the train one. Both sets were supplemented with label sets of corresponding sizes: N for the squeezed and N * S for the flattened set.

Obtaining embeddings for the train dataset texts (N = 45000) took roughly 180 minutes for the standard input model and 450 minutes for the extended input model.

The flattened set was created to evaluate the model's per-segment performance without averaging and was also required to analyze the embeddings and produce their visualizations. It was not used in any of the complete model implementations.

5.3.5 Final layer classifier

The classifier was implanted using a PyTorch Sequential model class. For the flattened dataset, a simple classifier with two hidden layers of sizes 16 and 2 linked with a ReLU activation function was used. For the squeezed dataset, adaptive average pooling was applied along dimension 0 in order to average the embeddings for all 30 input segments. After that, the averaged embedding was flattened to reduce the dimensionality and passed through identical hidden layers. The output of the model is therefore a tensor of size (1, 2) with the class prediction logits. This model was chosen as the final solution in our model and was used in the prediction pipeline that was passed to various explanation modules. It will be further referred to as FinalNetAvg. For the per-segment logits, two classifiers were used: one of them was a classifier with adaptive average pooling that used a 2-node hidden layer, while another one did not use any hidden layers at all and produced the output as the plain average of the 30 input logits. The latter, therefore, was not a model per se and did not require any training.

The classifiers with hidden layers were trained in a native PyTorch loop with *batch_size* = 30 using Adam optimizer and learning rate = 1e-4 selected using a learning rate optimization algorithm. The models were trained for 10 epochs, though peak values of the eval accuracy could already be achieved after 5 epochs. These findings do not quite agree with the experimental setup of Peng et al., in which the final classifier was trained for 400 epochs with a higher learning rate (a default Keras Adam optimizer learning rate of 1e-3). Without a consistent decrease in the training loss function score and improvement of the evaluation results, excessive training did not seem necessary.

5.4 Model results

Table 5.4 shows the results of our main model that has been chosen for further experiments compared to the original architecture of Peng et al. It can be noted that a gap of 0.9% exists for the F1 score. However, the scores fall between the two values reported by the authors: Val-1 scores for the test set of size 15780 and Val-2 for the test set of size 7601, which is being used in our evaluation as well.

Model	Binary accuracy	F1	AUC
Standard Input Bert + FinalNetAvg	0.933	0.936	0.933
Extended Input Bert CLS + FinalNetAvg	0.942	0.944	0.943
Peng et al. Val-2 bert4keras + Keras	N/A	**0.945**	**0.944**
Peng et al. Val-1 bert4keras + Keras	N/A	0.926	0.920
Naïve baseline	N/A	0.786	0.796

Table 5.4: *Performance of our model compared to the model of Peng et al. (2019 and the baseline*

Table 5.5 provides the comparison of the model of our choice with different architectures of the final classifier and different types of input embeddings – namely, produced by standard and extended input BERT models. Moreover, we compared the model's performance with the classifier layer receiving different inputs: embeddings of the last layer CLS token, sum of embeddings of CLS tokens from four last layers, and plain logits.

Model	Binary accuracy	F1
Standard Input Bert CLS + FinalNetAvg	0.933	0.936
Extended Input Bert CLS + FinalNetAvg	**0.942**	**0.944**
Extended Input Bert 4-layer CLS + FinalNetAvg	0.935	0.937
Extended Input Bert logits + 2-node Avg	**0.942**	**0.944**
Extended Input Bert logits + plain average	0.941	0.943

Table 5.5: *Performance of our model in different sizes and with different processing of embeddings*

It can be observed that, even though the growth in performance for the extended input model is still noticeable, it is considerably smaller for the averaged text-level prediction compared to the segment-level one: the growth of F1 score is only 0.8% as opposed to 2.5%.

The results also show that the model with the last-layer CLS token performs best (0.942/0.944) on par with the one classifying averaged logits, while the sum of logits shows a noticeably worse performance (0.935/0.937). For averaging logits, we tried both training a 2-node classifier and using plain averaging. Although the performance was very similar, plain averaging results were marginally lower by 0.1%, and some trained instances of the 2-node classifier reached even better results (up to 0.944/0.946), though such scores were irreplicable and were most likely due to overfitting on specific evaluation instances, and therefore were not further used for the final evaluation on the test dataset.

5.5 Model discussion

These results show that, in general, averaged logits can be used for classification as is, without training any additional dense layers, since the decline in performance is insignificant.

Considering the choice of the model size, the accuracy gain on the text level after increasing the size was noticeable (0.9%), but considerably smaller than that of single segments (2.9%). Given the dramatically longer training time (64 hours instead of 26) and the need to separate the segments from one text into two batches, the trade-off between time and prediction quality can be resolved differently depending on the goals of a particular experiment and on the type of target texts.

For longer texts, which can be separated into 30 or even more segments, using a faster and lighter model seems to be preferable, since averaging multiple predictions does not depend so strongly on the input size, and both training and generating embedding representation take significantly more time for the extended input model with an almost threefold increase. For smaller texts, particularly those that constitute only one segment, the choice can be made in favor of better quality of each prediction, since the options for averaging are limited and increasing the input size becomes the most plausible way to reach higher accuracy.

6. Final classifier interpretation

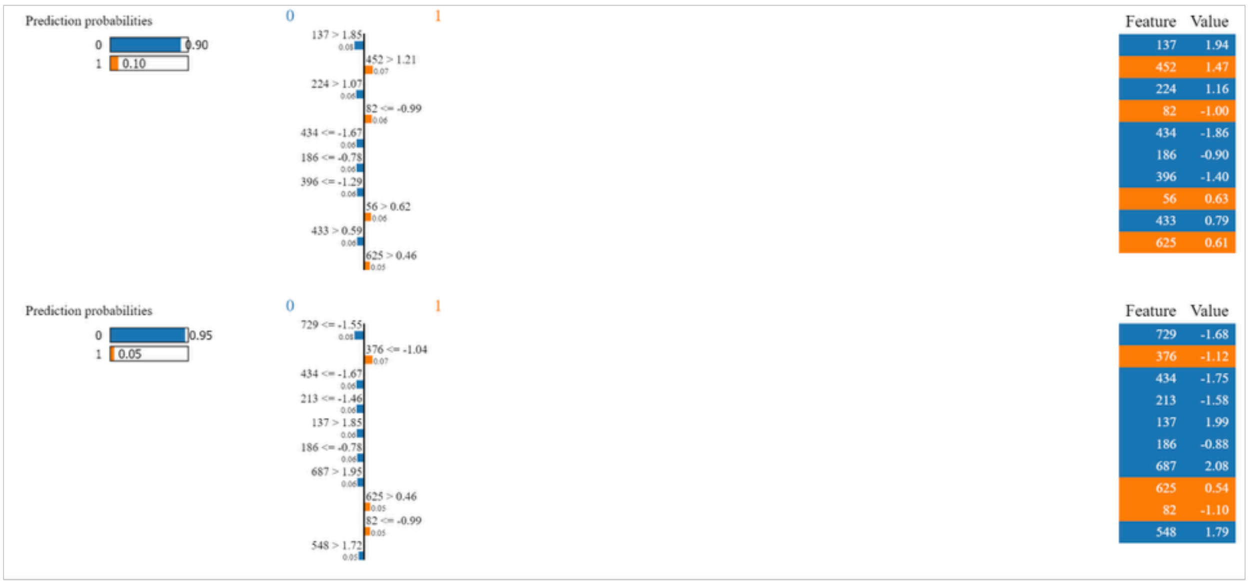
For generating explanations, we adopted a top-down approach, which assumes moving from the final layer of the model, which is the text-level classifier, down to the segment-level classifier and then to the internal structure of the BERT model, namely to the attention.

In this section, we use LIME explanations to analyze the behavior of the final classification model that uses the embeddings of segments as input and outputs class predictions. We then analyze the CLS embeddings that were obtained as the output of our fine-tuned BERT model and are now used as classifier input, observe how are they treated by the classifier and compare their saliency to the frequency of their appearing in LIME explanations.

As a result, we aim at answering whether the classifier in this setting is learning some new information to make its final prediction, or these predictions are already contained in the input embeddings as a result of fine-tuning, and by averaging the input from each sequence we can already provide a sufficiently accurate explanation.

6.1 LIME-tabular explanations overview

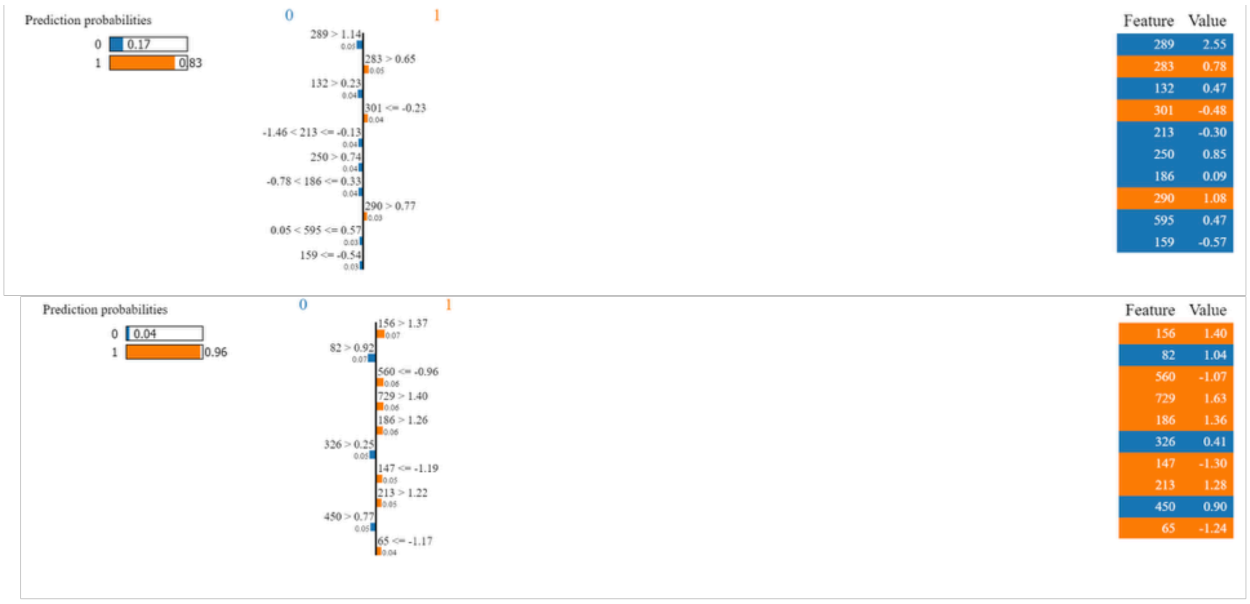
We used lime-tabular, a version of the LIME explainer that can accept two-dimensional input, such as the embeddings in question. Since the standard version of the classifier uses three-dimensional input ($N \cdot 30 \cdot 768$), we had to consider two separate types of explanations: those generated for a single pair of sequences (one of 30 elements along dimension 2) and the one using averaged output (where the mean value is calculated along dimension 2).



Feature	Value
137	1.94
452	1.47
224	1.16
82	-1.00
434	-1.86
186	-0.90
396	-1.40
56	0.63
433	0.79
625	0.61

Feature	Value
729	-1.68
376	-1.12
434	-1.75
213	-1.58
137	1.99
186	-0.88
687	2.08
625	0.54
82	-1.10
548	1.79

Figure 6.1: *Predictions for two adjacent segments (1/30 and 2/30)*



Feature	Value
289	2.55
283	0.78
132	0.47
301	-0.48
213	-0.30
250	0.85
186	0.09
290	1.08
595	0.47
159	-0.57

Feature	Value
156	1.40
82	1.04
560	-1.07
729	1.63
186	1.36
326	0.41
147	-1.30
213	1.28
450	0.90
65	-1.24

Figure 6.2: *Predictions for non-adjacent segment (10/30) and segment from another text with diff. label*

For single pairs of sequences, we generated explanations for several settings: two adjacent pairs of sequences, two pairs from one text, and two pairs from random texts. For each case, explanations were computed for the 10 most important feature-value combinations, which is the standard mode for lime-tabular. Due to a huge number of candidate features (768), the difference in weights between features in explanations is subtle. Weights over 0.1 were only observed sporadically, while in most cases feature #1 had a weight no bigger than 0.08. Nonetheless, it was observed that the top 10 most important features overlap in different explanations. The number of common features is highest for adjacent segments (50%), but even across non-adjacent segments and segments from texts with different labels, there was a number of overlapping features (2–3 in our observations). Also, some features may also maintain their relevance across segments, but with slightly lower importance, and therefore be excluded from the top 10. Examples of explanations are provided in Figures 6.1 and 6.2

The explanations for sequences averaged within one text were similar in terms of overlapping features, but the weights of the most important features were generally lower. Besides, we observed that the explanations differed in terms of confidence: while common confidence for separate segments was within 80–95%, the confidence of predictions for averaged segments was either even higher (close to 100%) or much lower, around over 55–60%. This can be explained by the fact that for some texts the segments yielded predictions with both labels, and averaging the embeddings of such segments, which likely contained opposite values, resulted in a decline of confidence, as well as in lower weights of features in explanations in general. If, on the contrary, the same labels were predicted for all segments, the final prediction could be made with higher confidence since the individual variance of each segment is mitigated by averaging.

6.2 LIME-tabular explanations analysis

To obtain a broader view and to uncover the extent of the top features' re-occurrence, we checked their distribution for multiple explanations. To that end, we generated 100 explanations for instances randomly sampled from the test data and collected the most popular features — that is, we counted the number of times a particular feature occurred among the 10 features that LIME extracted. We only considered the counts of the features rather than their values provided by explanations, since the subtle differences in these values would create too many unique feature-value combinations and hinder the understanding of this experiment.

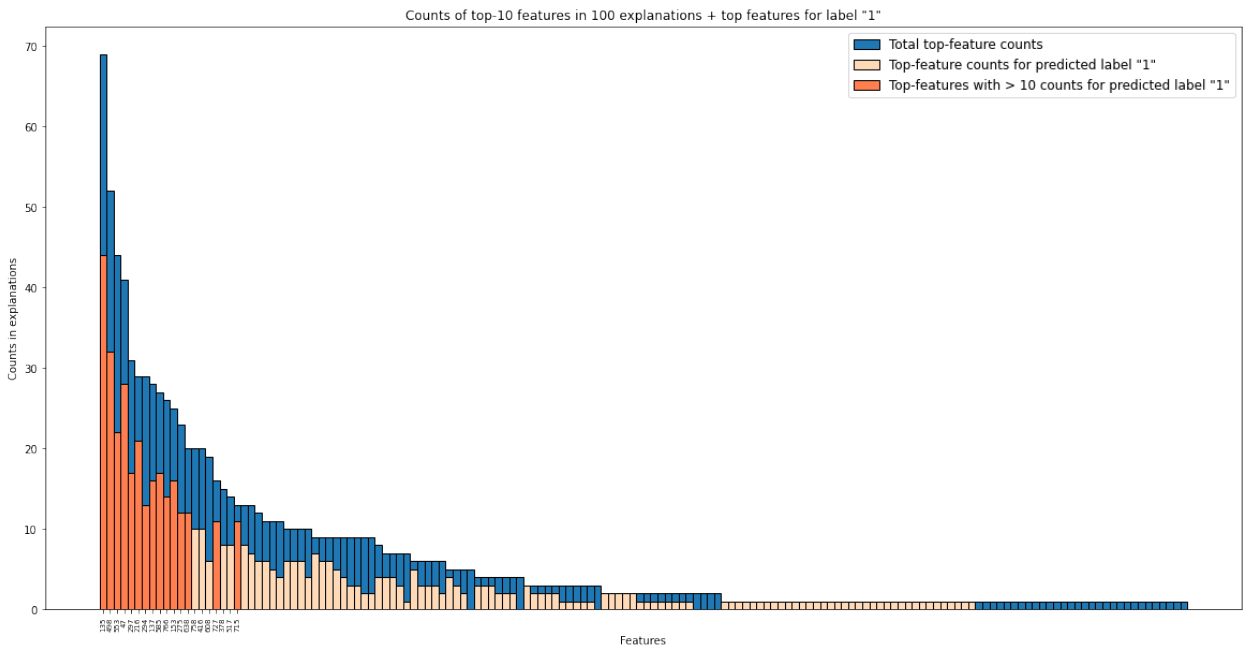


Figure 6.3: *Counts of features selected by LIME-tabular as top-10 for instances for label "1", with total counts for both labels in blue in the background*

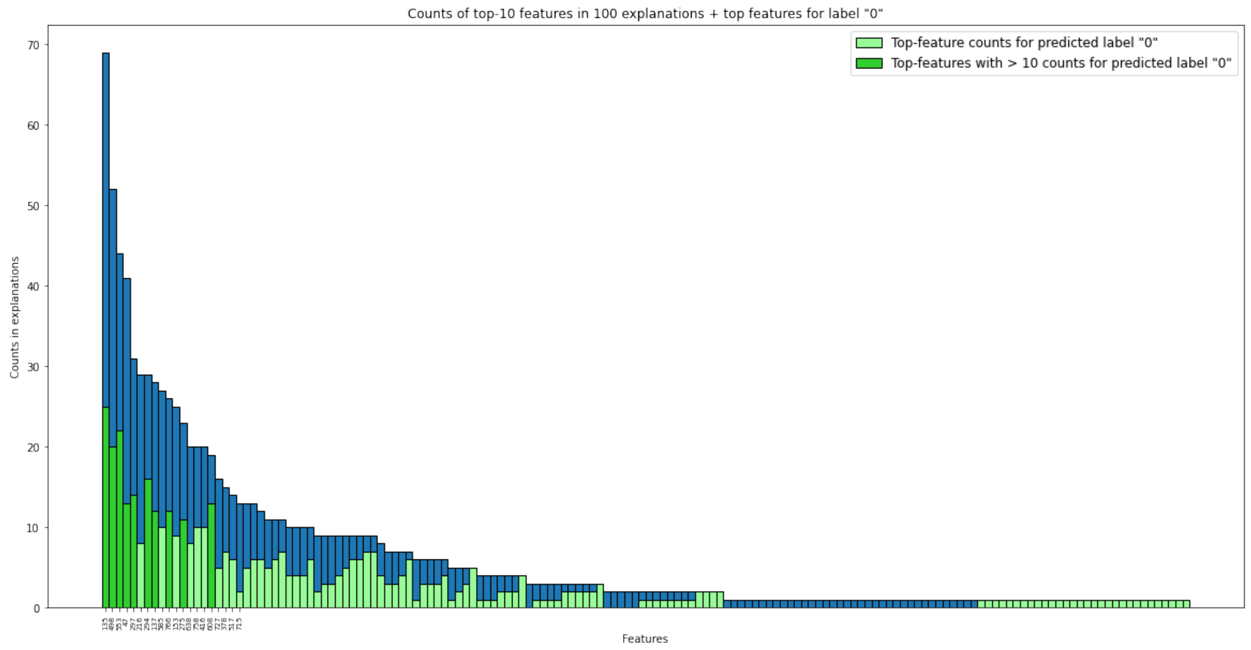


Figure 6.4: *Counts of features selected by LIME-tabular as top-10 for instances for label "0", with total counts for both labels in blue in the background*

We counted the occurrences for the whole sampled subset and separately for samples with the predicted label «1», with the predicted label «0», and with the erroneous predictions (where the predicted label is different from the true one).

In total, 165 features out of a total of 768 occurred as most important for LIME at least once. Given that in total LIME had to select 1000 features (10 for each of 100 explanations), this result suggests significant overlapping. Figures 6.3 and 6.4 show the distribution of counts for two classes separately, as well as the total counts. The most common features (>10 counts) are highlighted.

It can be seen that there is one most popular feature, *135*, that appears in almost 70% of explanations, and 15 features that appear in 20% or more. Their distribution between the two classes slightly differs: feature *47* (bar 4) is considerably more common for the class "1" than for the class "0". Feature *216* is also more important for label "1", while it is not even among the opposite class's most common ones. Features *294* and *608* are similarly more common for label "0" than for label "1".

Thus, we verified that a few top features are extremely common in explanations for both classes, which suggests that contrasting values of these particular features are crucial for the final prediction. We also discovered that there are also some atypically common for one particular class, or for misclassified segments. However, at that point, there was no observable link between the importance of classifier input features and interpretations of those features with respect to the input texts.

6.3 CLS embeddings analysis

In search of a link between the classifier's performance, which was partially uncovered by LIME, and the characteristics of BERT embeddings produced by our model, we analyzed the difference between the embeddings across two classes. We then compared the results for the training and the test sets and contrasted them to the embeddings produced by a standard BERT model without fine-tuning. Finally, we calculated the Pearson correlation coefficients between the embeddings and the weights learned by the classifier. We aimed at determining whether the features most frequently occurring as most important ones for the classifier are also those that are the most salient in the input embeddings.

Figure 6.5 shows the weights of features ranked according to their number, averaged for each class separately. It can be seen that the values are typically opposite for large groups of features: a positive average value for class 1 is frequently accompanied by a negative value for class 0 and vice versa.

The average weights of features in embeddings produced for the Test set, shown in Figure 6.5 (bottom), are largely similar to those of the Train set (top): the groups of positive and negative values can be observed. However, the absolute values tend to be smaller. This can be explained by the fact that a higher number of prediction errors compared to the training set results in both positive and negative values being averaged as part of the same class, thus decreasing the absolute value.

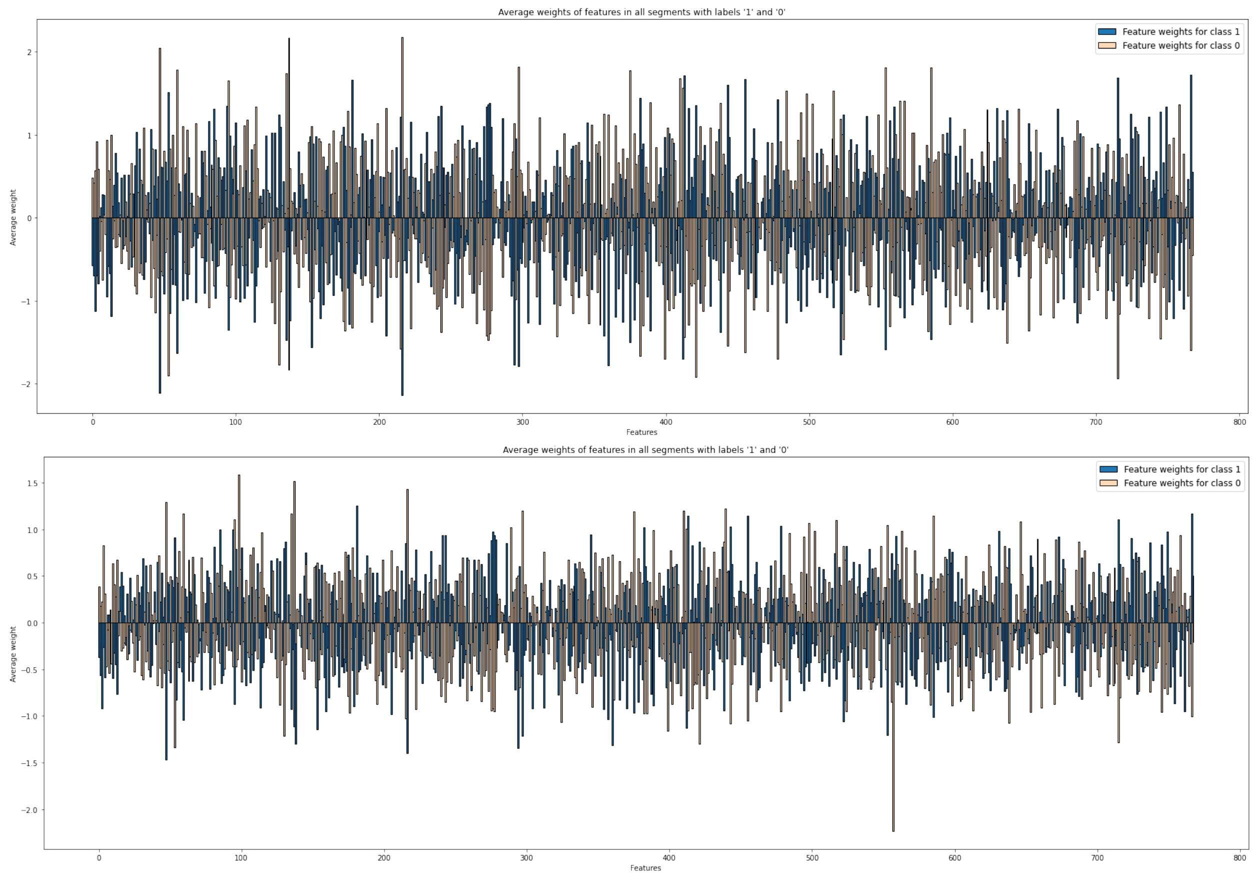


Figure 6.5: *Average weights of features in all segments with labels "1" (blue) and "0" (gold) in Train subset (above) and Test subset (below)*

After sorting the feature weights by value, it can be observed in Figure 6.6 that within one class features with different values are generally distributed evenly, with a similar increment in weight, excepting a small number of both positive and negative features with the highest absolute values.

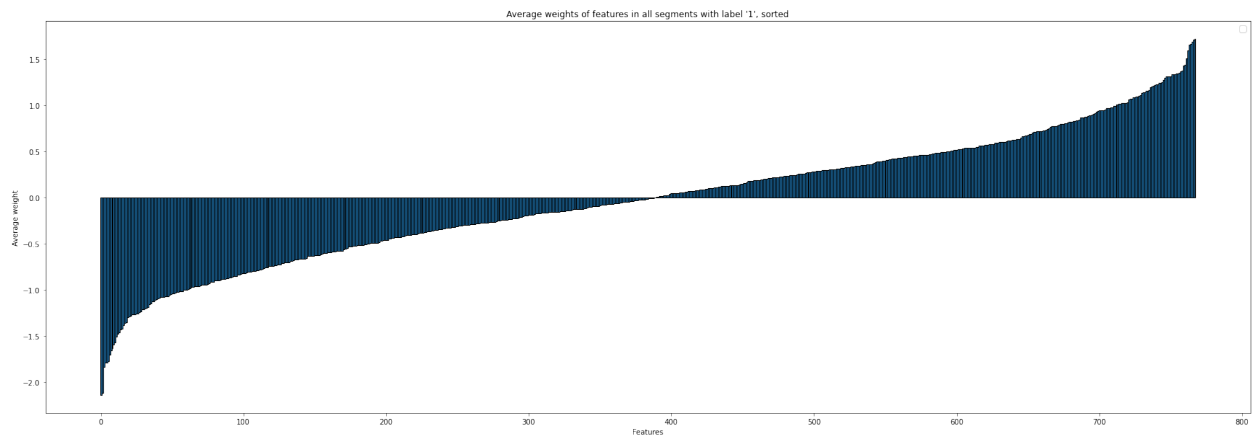


Figure 6.6: *Average weights of features in all segments with labels "1" for the Train subset, sorted by values*

To see how these features enable discriminating between the two classes, we considered the absolute values of the difference between the averaged weights (Figure 6.7). A somewhat similar picture was observed: a comparatively small subset of approximately 50 features showed a noticeably larger increment in absolute values of differences than those following them.

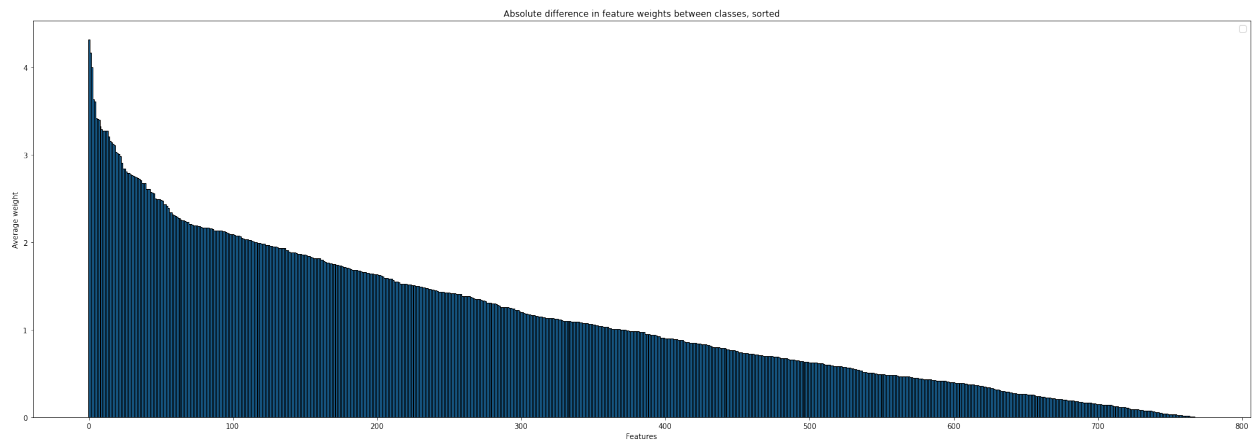


Figure 6.7: *Differences between average weights of features between classes "1" and "0" for the Train subset, sorted by absolute values*

6.4 Comparison of LIME-tabular and difference-based features

In order to observe whether such salient differences are taken into account by the classifier, we calculated the overlap between these features with highest difference between weights in different classes and the list of most frequent features extracted by LIME in the previous experiment. The results are summarized in Table 6.1.

For LIME-tabular explanations of 100 items sampled from the Train dataset, we selected the 50 most frequently extracted features and compared this list with 50 features that have the most salient weight difference between classes in embeddings produced for the Train dataset. We discovered that 52% of features from these lists were overlapping. After following the same procedure for LIME explanations of samples from the Test dataset, we found 50% of overlapping features. We repeated the experiment with the embeddings of the Test dataset and obtained the result of 54% and 48% respectively. It can be noticed that this percentage of overlapping features is considerably large given the total number of candidate features (768) and the limited sample (50). Indeed, over half of the 50 features most commonly chosen as most important ones are also among those with the highest difference between classes. Also, for both Train and Test dataset embeddings, the overlap is slightly larger with LIME features from explanations of instances sampled from the Train dataset. It suggests that the classifier's performance on Train data is slightly more consistent with the initial information represented in the embeddings.

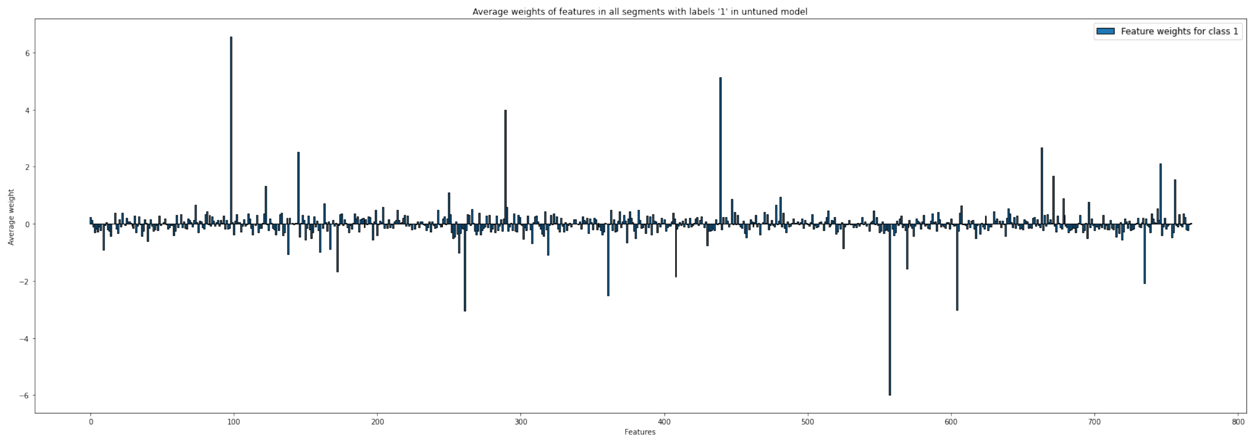
To ensure that such behavior is justified by the model's performance and that the difference between classes is indeed learned at the pre-training stage, we also calculated the overlap for the embeddings produced by the untuned BERT model. As before, 50 features with the highest absolute difference between classes were selected. In this case, however, the percentage of overlapping features was significantly smaller: 10% were overlapping with LIME features on test samples, and a mere 6% — with those on train samples.

	LIME for Train set	LIME for Test set
Train set embeddings (fine-tuned model)	52 %	**50 %**
Test set embeddings (fine-tuned model)	**54 %**	48 %
Train set embeddings (untuned model)	10 %	6 %

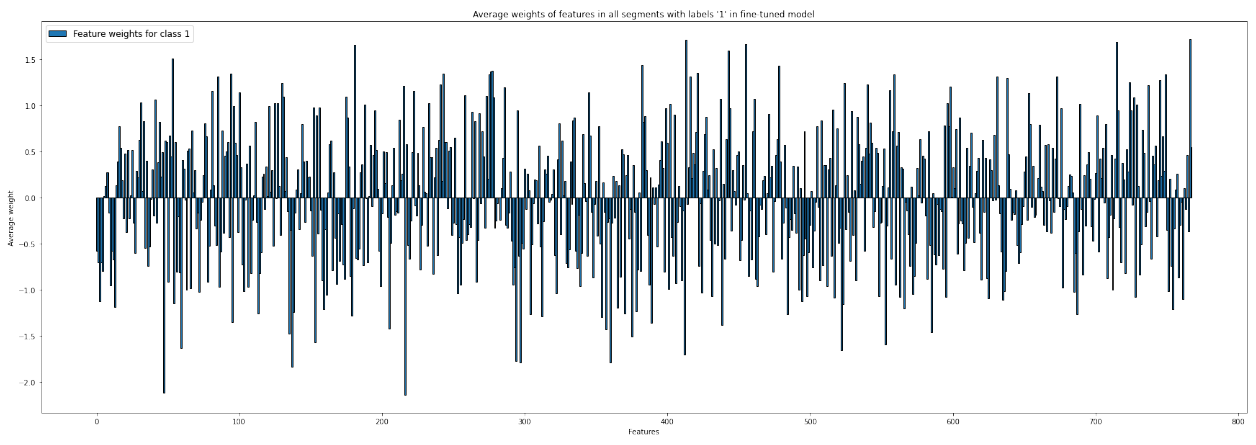
Table 6.1: *Overlap between embedding components with the highest difference between classes and embedding features extracted by LIME for Train and Test subsets in fine-tuned and untuned models*

6.5 Untuned CLS embeddings analysis

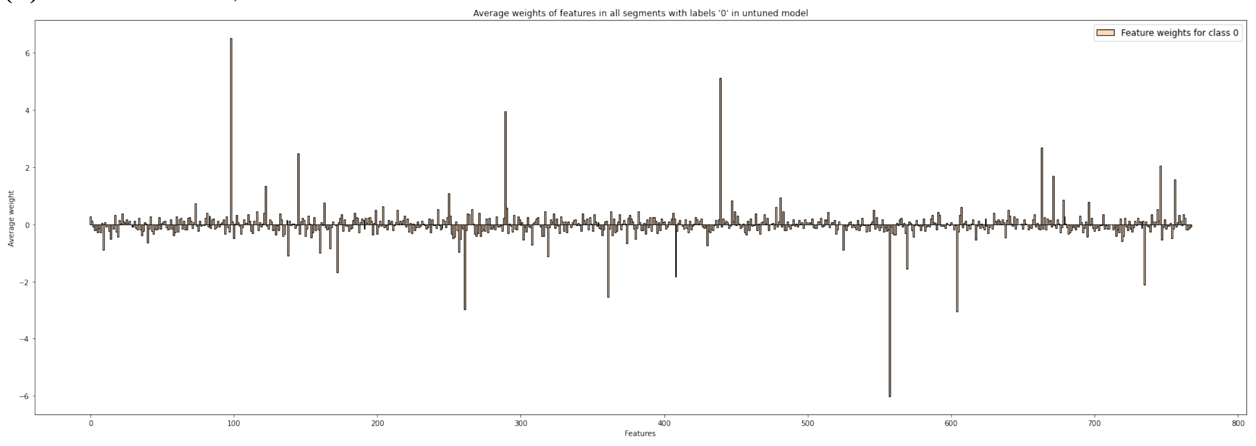
The results suggest that untuned embeddings differ significantly in terms of feature weights. To verify it, we analyzed them in the same way as before. Figure 6.8 shows that feature weights for two classes are almost identical, which is quite the opposite picture compared to fine-tuned embeddings that had contrasting values. Besides, most average weights are close to 0, which suggests that a single class included both positive and negative values.



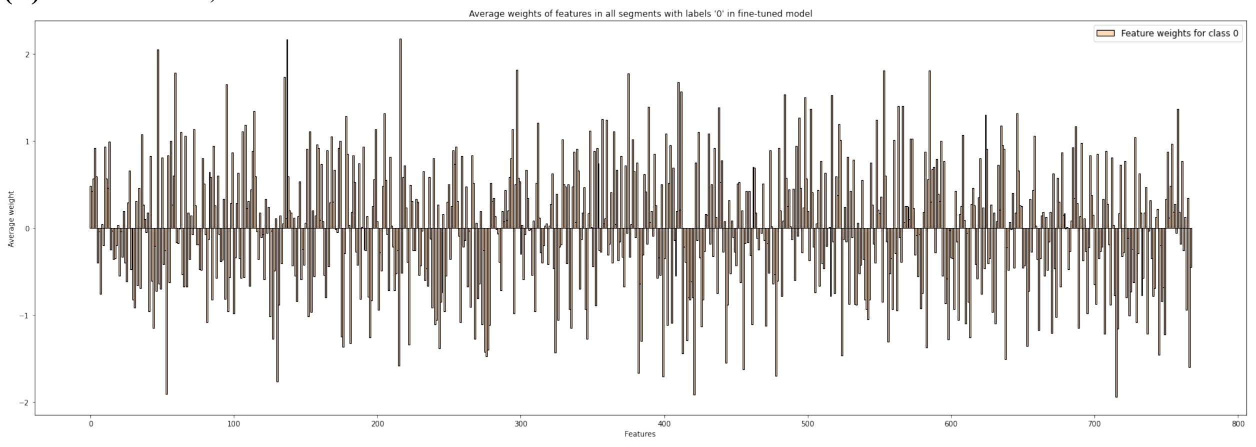
(A) *Untuned model, label "1"*



(B) *Fine-tuned model, label "1"*



(C) *Untuned model, label "0"*



(D) *Fine-tuned model, label "0"*

Figure 6.8: Average weights of features in all segments with labels "1" (A, B) and "0" (C, D) in Train subset for the untuned (A, C) and fine-tuned model (B, D)

Moreover, even though some differences in absolute values between classes exist, they are dramatically lower than those in fine-tuned embeddings, where the most salient difference is roughly 20 times bigger, as can be seen in Figure 6.9.

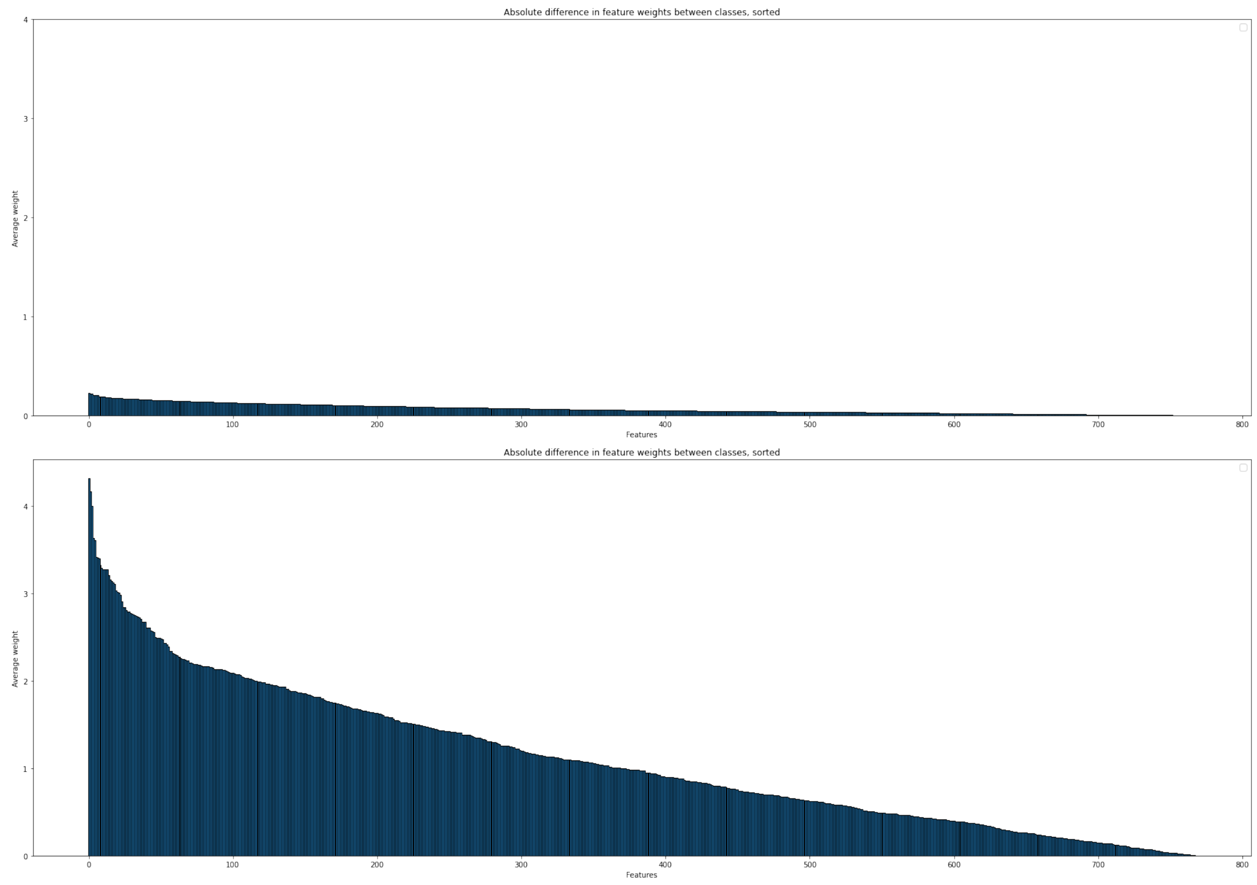


Figure 6.9: *Differences between average weights of features between classes "1" and "0" for the Train subset, sorted by absolute values, for untuned model (above) and fine-tuned model (below)*

6.6 Correlation between CLS embeddings and classifier nodes

Generating multiple LIME explanations for sampled instances provided some understanding of the classifier as it showed that a limited number of features is regularly used in predictions for both classes. Comparing the averaged values of features in embeddings for each class showed that these values are primarily opposite. To fully grasp the behavior of the classifier, we needed to see how these values are processed in the hidden layers. The model's architecture included two fully-connected layers of sizes 16 and 2 respectively, connected by a ReLU activation function. Two nodes of the second layer correspond with the final prediction, as they are turned into class probabilities after the softmax activation. Therefore, we expected the first layer to be capable of selecting features responsible for each class.

To visualize the relations between the nodes, we plotted a heatmap of Pearson correlation coefficients between the parameters of all 16 nodes in Figure 6.10. To show how they correspond to embeddings of different classes, we also added the averaged feature weights for class 1 (C1) and class 0 (C0) in fine-tuned Train embeddings. These weights indeed show a strong negative correlation (-0.93) which goes in line with the previous observations in Figures 6–7. Besides, the nodes turned out to be divided with respect to correlation: nodes 1–4, 7, and 15 show a positive correlation with class 0, while other nodes – with class 1. In C0 and C1, the positive correlation is generally stronger, and it reaches 0.76 for node 4 in C0.

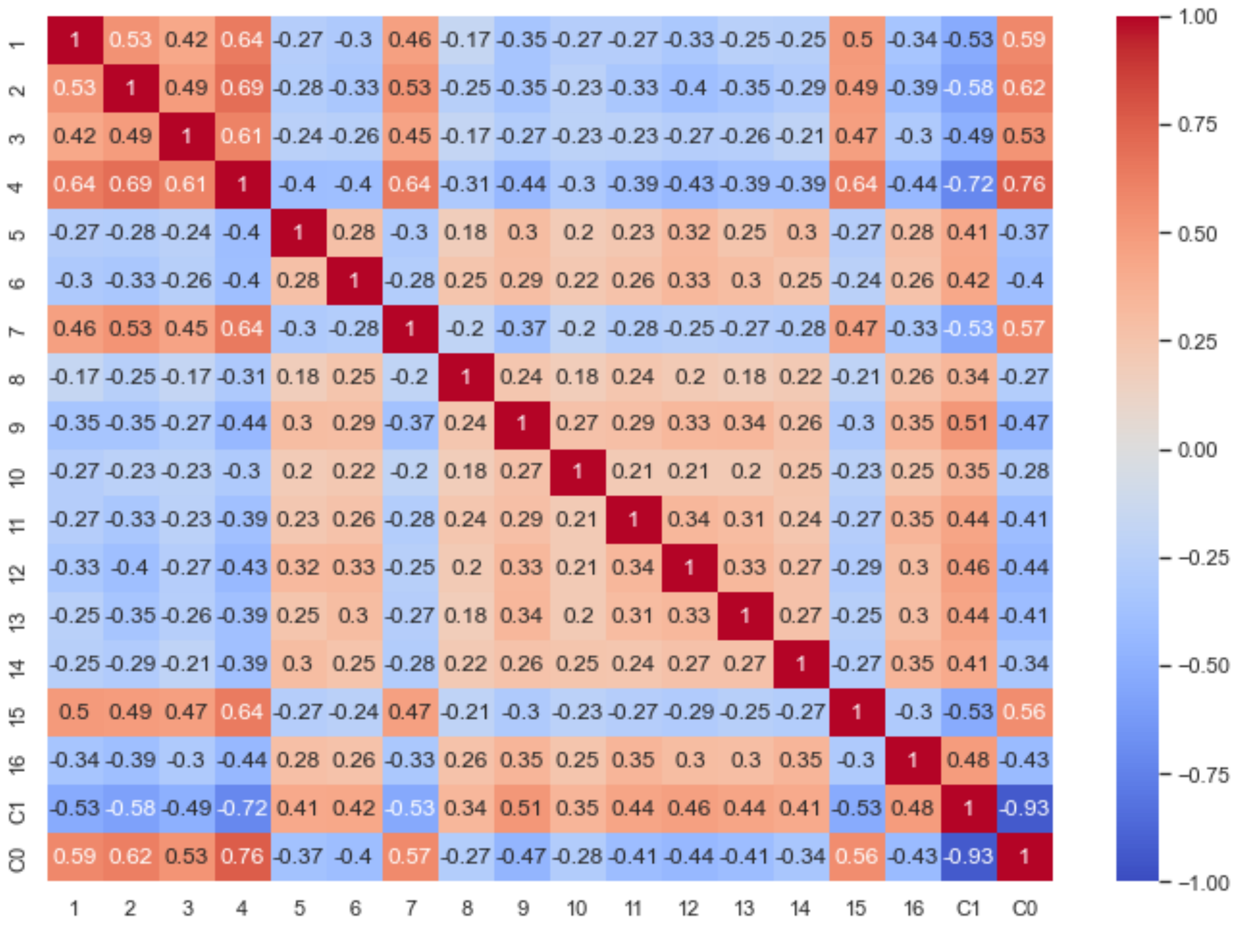


Figure 6.10: *Pearson correlation between weights in 16 nodes in the first fully-connected layer of the classifier (1–16) and averaged embeddings of class "1" (C1) and of class "0" (C0)*

Given the ReLU activation function that immediately follows the first fully-connected layer and sets all negative values to zero, the only way in which the model can maximize the values of the desired class is by multiplying positive values of relevant features by positive parameter values, and negative ones – by negative parameter values, while leaving irrelevant features negative or close to zero. Thus, a positive correlation between the class and the node parameters suggests that the node in question selects instances with feature values relevant to that class.

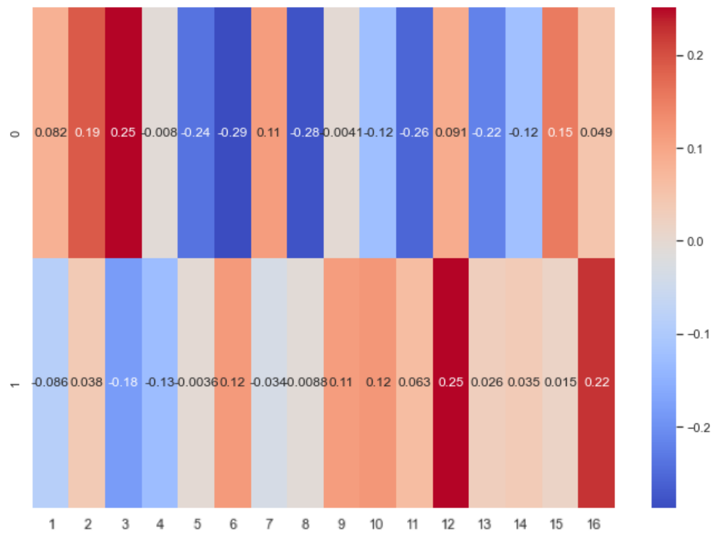


Figure 6.11: *Pearson correlation between weights in 2 nodes in the second fully-connected layer of the classifier (0–1)*

The parameters of 16 nodes in the second fully-connected layer were also visualized with a heatmap in Figure 6.11. Some traces of the division observed in the correlation heatmap can also be found here: nodes 1–4, 7, and 15, which had a positive correlation with class 0, are assigned higher weights in this layer's node 0.

6.7 Classifier layer discussion

These findings indicate that the classifier essentially uses its first fully-connected layer to emphasize the high values of features relevant to a particular class. After that, for most nodes, it further increases the difference between positive and negative values by applying positive weights to the nodes that show a positive correlation with a class. As a result, a high positive value is achieved for the desired class and a low negative one – for the irrelevant class.

It seems that such a procedure does not necessarily require a separate model with hidden layers. Given that the encodings of each sequence pair already show contrasting values for different classes and that that contrast can be preserved after averaging the sequence pairs from one text, we could either use a classifier without averaging or consider the logits of the fine-tuned BERT instead of the CLS token embeddings, since they can also represent the contrast encoded in these embeddings.

Therefore, it is possible to disregard further explanations of the joint model including both BERT and the final classification layer and to proceed to the explanations of fine-tuned BERT individually. Given that the predictions for 30 segments are plainly averaged without any substantial transformations, we are now able to consider the explanations for separate segments. A joint explanation can thus be created, if needed, by considering all segments or by providing example explanations for some, most expressive, cases.

7. LIME explanations for segments

The results of the previous chapter enabled us to conclude that the contribution of the final classifier applied to the CLS embeddings of all segments boils down to discriminating between two sets of almost opposite embeddings that are shaped by the fine-tuned BERT, and no additional transformation is performed in the hidden layers. This reasoning is further verified by a merely marginal drop in performance when plain averaging is used instead of a classifier with two dense layers.

These considerations enabled us to move from text-level explanations to a smaller scale and concentrate on segment-level explanations, given that further changes in the prediction that are happening at the final classification step are fully interpretable and reduced to averaging. Joint explanations can thus be created, if needed, by considering all segments or by providing example explanations for some most expressive cases.

In this chapter, we introduce the original LIME tool for generating explanations and analyze its performance on segment-level predictions for both classes. We then reflect on the results, suggest an explanation for the difference in performance between classes, and propose an updated model that may be used to mitigate this gap. Finally, we compare the results of the original and the updated models on different examples, probe the correctness of assigned weights and the robustness of classification and use the outcomes for outlining the potential set of features involved in classification.

7.1 LIME overview

As the primary means of generating initial explanations, we have chosen the original implementation of LIME (Ribeiro et al., 2016)). LIME is a model-agnostic technique of post-hoc explanations, which implies that it can be used regardless of the inner structure of the model, only considering input and output, and that explanation is happing after the prediction is obtained, using additional operations (Danilevsky et al. (2020).

In LIME, a local surrogate model is learned for approximating the model's behavior in case of a particular prediction. To that end, LIME creates some "perturbed" samples by randomly removing some subsets of data (random lists of words, in case of LIME-text explainer) and supplementing the resulting input with cosine distance from the original text as the default metric, so that higher weight is assigned when perturbed input is close to the original, but prediction for it differs substantially.

The tool performs a large number of random permutations (5000 by default). A locally linear sparse model is learned based on this perturbed data. That proxy model is then interpreted by highlighting the contribution of the most important features to its prediction. The weights of these features should approximate the original model's behavior: removing each of these features should shift the prediction towards the opposite class by this weight, and this property is ideally supposed to be additive.

7.2 LIME-text for attribution explanations

In this section, we try to extract candidate important features that can be shown to influence the prediction and therefore correlate with the classifier's features.

To do so, we use LIME-text to explain the complete pipeline of our model, as well as the pipeline of the naïve baseline.

The main prerequisite of a properly functioning LIME explanation is a classification model pipeline that satisfies the following requirement: it has to be able to accept a list of single texts as input and produce a Numpy array of probabilities (logits processed through the softmax layer) as output. For that reason, we implemented different types of pipelines: for pairs of segments, for pairs of texts, and for a single input that combines both texts from a pair with a separator.

The explainer for textual input only accepts a single text, while our model is trained to process a pair of texts. Therefore, the input pair in the LIME-compatible pipeline is initially combined with a separator made of special characters (so that it is not influenced by word-level perturbation, as would be the case with the usual SEP token used in BERT) and then split back at the preprocessing stage.

Perturbations can be performed in two modes: bag-of-words (default) and non-bag-of-words. In the latter, the words are randomly removed based on their positions, while in the former perturbations are performed on the vocabulary of the text so that all occurrences of a selected word are removed. For this task, we utilized the standard bag-of-words mode, given that the alphabet of features would otherwise be too large.

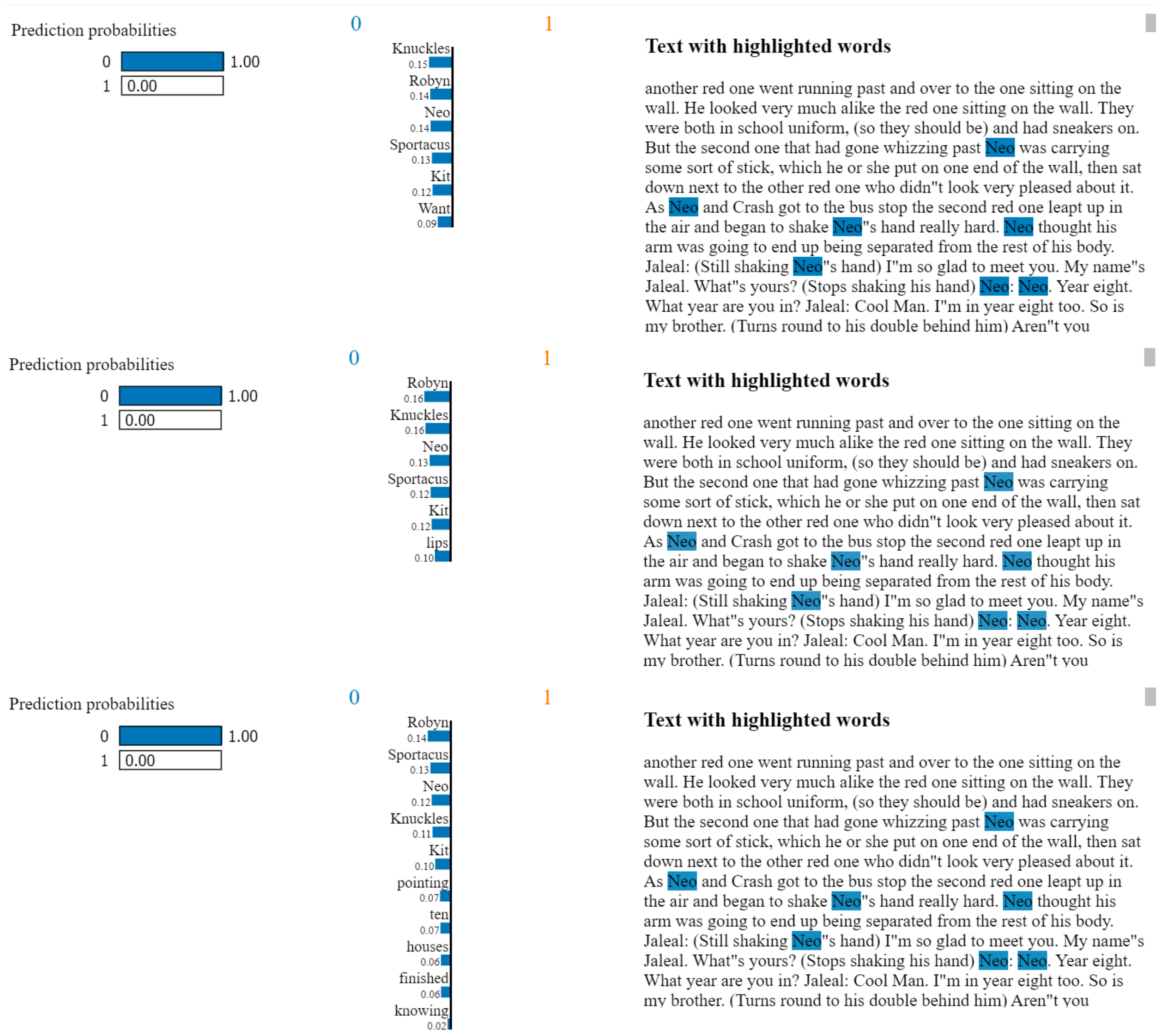
7.2.1 Class 0 explanations

The first explanation, shown in Figure 7.1 (top), was generated for a single pair with label 0 ("not the same author"), correctly predicted. This prediction shows very high confidence (p=1.0). Generating an explanation with top-6 features for the whole model took about 13 minutes, compared to 3 minutes for the embeddings classifier, with standard 5000 perturbations. All of the top-6 features shift the decision towards the correct class prediction and explain a total of ~77% of the prediction, which is considerably different from the embeddings classifier, in which top-10 features explained roughly 60%.

All of those features seem to be meaningful, though different in nature. Five of them are in fact character names from both texts (*Knuckles* and *Neo* only appear in the first fragment, while *Robyn*, *Sportacus*, and *Kit* — only in the second one). It may seem that in this example the model primarily functions as a named entity recognizer, but the dataset is constructed with cross-topic classification in mind: both same-author and different-author pairs include texts from different fandoms (therefore, with at least some different names), and same characters, in turn, may appear in pairs with both labels.

The 6th feature is the word *Want*, which is indeed a suitable candidate parameter for style formalization. This is a frequently used verb that has a significant amount of contextual synonyms and stylistic variants due to its broad meaning that sometimes expresses modality. Thus, an author can freely choose between *want* and its alternatives without changing the meaning, which makes it a good indicator of individual stylistic footprint. In this example, the counts of the word *want* differ between the two texts: out of 18 occurrences, only 2 are in the first text. According to LIME, such stylistic dissimilarity adds 9.3% to the probability of these texts being written by different authors.

We then generated more explanations for the same data with top-6 features and top-10 features, also shown in Figure 7.1 (middle and bottom). The 5 most frequent features re-appeared in all explanations with but slight re-arrangement. The situation with other features was different: these were different words that appeared in dissimilar proportions. The word *pointing* appeared in a proportion close to that of *want*, but the words *houses*, *lips*, and *ten* only appear once. It could be concluded that the top-5 features are followed by a longer tail of roughly similarly important features.



Text with highlighted words

another red one went running past and over to the one sitting on the wall. He looked very much alike the red one sitting on the wall. They were both in school uniform, (so they should be) and had sneakers on. But the second one that had gone whizzing past Neo was carrying some sort of stick, which he or she put on one end of the wall, then sat down next to the other red one who didn"t look very pleased about it. As Neo and Crash got to the bus stop the second red one leapt up in the air and began to shake Neo"s hand really hard. Neo thought his arm was going to end up being separated from the rest of his body. Jaleal: (Still shaking Neo"s hand) I"m so glad to meet you. My name"s Jaleal. What"s yours? (Stops shaking his hand) Neo: Neo. Year eight. What year are you in? Jaleal: Cool Man. I"m in year eight too. So is my brother. (Turns round to his double behind him) Aren"t you

Text with highlighted words

another red one went running past and over to the one sitting on the wall. He looked very much alike the red one sitting on the wall. They were both in school uniform, (so they should be) and had sneakers on. But the second one that had gone whizzing past Neo was carrying some sort of stick, which he or she put on one end of the wall, then sat down next to the other red one who didn"t look very pleased about it. As Neo and Crash got to the bus stop the second red one leapt up in the air and began to shake Neo"s hand really hard. Neo thought his arm was going to end up being separated from the rest of his body. Jaleal: (Still shaking Neo"s hand) I"m so glad to meet you. My name"s Jaleal. What"s yours? (Stops shaking his hand) Neo: Neo. Year eight. What year are you in? Jaleal: Cool Man. I"m in year eight too. So is my brother. (Turns round to his double behind him) Aren"t you

Text with highlighted words

another red one went running past and over to the one sitting on the wall. He looked very much alike the red one sitting on the wall. They were both in school uniform, (so they should be) and had sneakers on. But the second one that had gone whizzing past Neo was carrying some sort of stick, which he or she put on one end of the wall, then sat down next to the other red one who didn"t look very pleased about it. As Neo and Crash got to the bus stop the second red one leapt up in the air and began to shake Neo"s hand really hard. Neo thought his arm was going to end up being separated from the rest of his body. Jaleal: (Still shaking Neo"s hand) I"m so glad to meet you. My name"s Jaleal. What"s yours? (Stops shaking his hand) Neo: Neo. Year eight. What year are you in? Jaleal: Cool Man. I"m in year eight too. So is my brother. (Turns round to his double behind him) Aren"t you

Figure 7.1: *Text-level LIME features, true class "0", bag-of-word setting, for different runs of LIME-base with top-6 features (top and middle) and top-10 features (bottom) selected. Only an excerpt of the text is visualized*

However, the distribution of importance in top-10 features doesn't support this view: the importance vanishes quite rapidly, with feature 10 being 3 times less important than feature 9 (0.06 → 0.02). Top-9 features in this case explain 86% of the prediction and don't really leave any room for the high estimated importance of *want* (0.09) or *lips* (0.1).

We arrived at two possible explanations for that:

the local approximations can be inconsistent, and while alterations of ±0.2 are not so important for most common features, they significantly change the distribution of less common ones

some features may not be reduced to words themselves, and since LIME is limited to word-level explanations, it is bound to be uncertain about their importance.

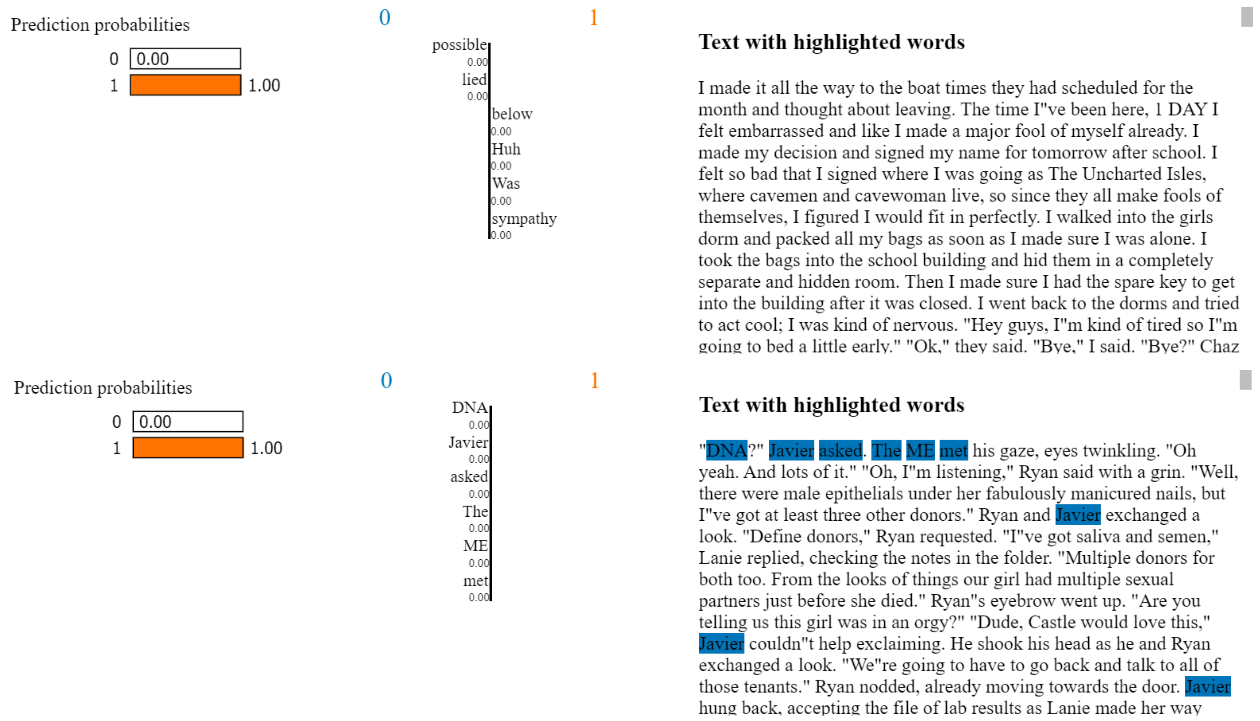
Nonetheless, all extracted features in this example seem to be meaningful in terms of explainability since they appear in two texts with different frequency. Therefore, removing them makes the texts more similar and decreases p of class 0.

7.2.2 Class 1 explanations

In explanations of texts for Class 1, shown in Figure 7.2, we can see that, despite the very high confidence of the model, LIME cannot explain its prediction at the word level. Top-6 features only marginally influence the prediction, with the highest value being a mere -2.85e-06. According to LIME, none of the word-level features can explain the model's confident prediction.

In the bottom example in Figure 7.2, we see that all extracted features point at class 0. though with extremely low weight, contrary to the actual prediction, and the top features are in fact just the first words from the text.

We observed similar behavior for most examples with label 1, and in all of them, the weights assigned to features were close to zero, and the intercept value of the local linear model was around 0.99.



I made it all the way to the boat times they had scheduled for the month and thought about leaving. The time I"ve been here, 1 DAY I felt embarrassed and like I made a major fool of myself already. I made my decision and signed my name for tomorrow after school. I felt so bad that I signed where I was going as The Uncharted Isles, where cavemen and cavewoman live, so since they all make fools of themselves, I figured I would fit in perfectly. I walked into the girls dorm and packed all my bags as soon as I made sure I was alone. I took the bags into the school building and hid them in a completely separate and hidden room. Then I made sure I had the spare key to get into the building after it was closed. I went back to the dorms and tried to act cool; I was kind of nervous. "Hey guys, I"m kind of tired so I"m going to bed a little early." "Ok," they said. "Bye," I said. "Bye?" Chaz

"DNA?" Javier asked. The ME met his gaze, eyes twinkling. "Oh yeah. And lots of it." "Oh, I"m listening," Ryan said with a grin. "Well, there were male epithelials under her fabulously manicured nails, but I"ve got at least three other donors." Ryan and Javier exchanged a look. "Define donors," Ryan requested. "I"ve got saliva and semen," Lanie replied, checking the notes in the folder. "Multiple donors for both too. From the looks of things our girl had multiple sexual partners just before she died." Ryan"s eyebrow went up. "Are you telling us this girl was in an orgy?" "Dude, Castle would love this," Javier couldn"t help exclaiming. He shook his head as he and Ryan exchanged a look. "We"re going to have to go back and talk to all of those tenants." Ryan nodded, already moving towards the door. Javier hung back, accepting the file of lab results as Lanie made her way

Figure 7.2: *Text-level LIME features, true class "0", bag-of-word setting, for different texts. Only an excerpt of each text is visualized*

7.2.3 Analysis of explanation difference

Interpretations of our model for class 0 are meaningful both in terms of their capacity to approximate the probability and their interpretability as linguistic features. This can be explained by the fact that LIME perturbations are capable of controlling the degree of dissimilarity between texts, which, ultimately, is the generalization of all the features used for capturing stylistic differences.

Indeed, if a certain word has a different frequency in two texts, we *decrease their dissimilarity* by removing it. If a corresponding change towards class 1 in the prediction happens, it proves that this word contributes to class 0 and makes the texts more dissimilar. The most salient features, in this case, are unique words (names and some singular occurrences) and the words substantially different in frequency.

Interpretations for class 1, however, don't seem to be meaningful and do not provide weights that account for the prediction. The likely reason for this is as follows: the goal of the explanation for class 1, in the end, is to find words without which the texts will become *less similar*, which would imply that these words make texts stylistically more similar. And this goal, as it turns out, is unachievable in this classical LIME perturbations scenario: we cannot perform such perturbations that would make texts less similar.

In case of removing a word feature that accounts for dissimilarity (a word with a different frequency in two texts), the texts will become more similar, which means that such perturbation can only increase the probability of class 1 — thus, such word cannot account for the probability of class '1'.

If we remove a word feature that accounts for similarity (a word with similar frequency in two texts), both texts are changed in a similar way; thus the degree of their similarity doesn't shift substantially, and such changes also cannot account for the probability of class 1.

Therefore, we claim that the degree of dissimilarity (in dissimilar texts) is a parameter controllable through classical LIME perturbations, but the degree of similarity (in similar texts) is not.

We formalized the problem of explanations for class 1 as a list of intermediate hypotheses.

Firstly, the problem may be in two texts, for such a setup enables manipulating dissimilarity, but not similarity. If this is the case, a version of LIME has to be created that can permute features of two texts independently: interpreting class 0 could be performed by removing dissimilarity when a word with dissimilar frequencies is deleted. Class 1 explanations, on the contrary, requires reduction of similarity, and we can do so by removing the equally distributed word from the first and the second text to see how important such equality is for the model.

Secondly, it may be the case that the model doesn't learn similarity features at all. It only learns what makes the texts dissimilar, and in the absence of such features, the default class is 1 ('same author'). By removing features from a text without dissimilarity, we have no means to access the features that are already absent given the a priori similarity of these texts. In this case, we can only use class 0 for explanations.

We aim at probing the first hypothesis to verify that explanations can be generated by performing a different type of input perturbation and that there exist such features that are responsible for classifying text pairs with label "1". If such explanations cannot be achieved by influencing the perturbation strategy, such result would suggest that the second hypothesis is the case and the model treats one of the classes as default.

7.3 Pair-aware perturbations in LIME

As mentioned above, LIME-text uses two modes. Therefore, we implemented two ways of independent perturbation of input.

7.3.1 Non-bag-of-words perturbation

For the non-bag-of-words mode, we enabled the Explainer to memorize the position of the separator and change the input in three different modes: "left" (only permute the first text in the pair), "right" (only the second one), and "random" (for each permutation, the side which is going to be altered is chosen randomly). The motivation behind the introduction of the "random" mode is that, even though on average both texts are involved equally, each particular perturbation is still performed on a single text, thus allowing a liner model to learn from a set of cases in which only one text is manipulated, and similarity that exists between the texts is therefore broken.

The "random" mode showed the results that are closest to the original LIME in the non-bag-of-words setting. The other two modes in the non-bag-of-words setting still could not produce meaningful interpretations. The most likely explanation for that is the large number of candidate features that the explainer faces when each word is treated individually and with its position in mind. For the set of 256 features, with some of them likely correlated (such as quotes, which are paired), creating a local linear approximation might be too complicated.

To reduce the number of candidate features and also obtain a more diverse set of important features when the explanation is generated, we proceeded with a beg-of-words mode.

7.3.2 Bag-of-words perturbation

For the bag-of-words setup, we had to adopt a different approach. The original LIME first builds the vocabulary for the input text and then selects random entries from that vocabulary and turns them off in the input text. In such a case, plainly limiting the scope of perturbations is not feasible because the randomly selected vocabulary entry may not be present in the text, and hence the average intensity of perturbations will decline, and so will the explainability of the linear model.

An additional limitation is that the perturbation cannot be edited or partially reconstructed after the initial application because it is performed together with the cosine similarity calculation. Changing the input afterward would impair the correctness of the similarity metric, which is crucial for correctly estimating the importance of perturbations.

Therefore, we had to build separate vocabularies for each of the texts, sample candidates for permutations from these vocabularies separately, and then apply them with the scope of each text, defined by the separator position, in mind. The methods for reconstructing the text from tokens for visualizing the explanation and for highlighting the words according to their estimated importance also had to be changed, taking the two vocabularies into consideration. All of the main classes had to be redefined in order to enable the preservation of the separator position and the current text from the pair which is being processed.

Given the global character of required changes, we did not implement a "random" mode for the bag-of-words setting: the scope of perturbations have to be chosen upon the initialization or the explanation. Consequently, the appropriate method of feature extraction with this version of LIME implies the application of explainers in both "left" and "right" modes and combining the lists of features extracted by both of them, taking the importance of these features into account.

Application of the bag-of-words explainer for pair-aware input alternation finally resulted in explanations in which the weight of the extracted features did not approach zero, and the intercept weight of the local linear model approximating the prediction was lower than 0.99 (as was the case with the original LIME), commonly not exceeding 0.3.

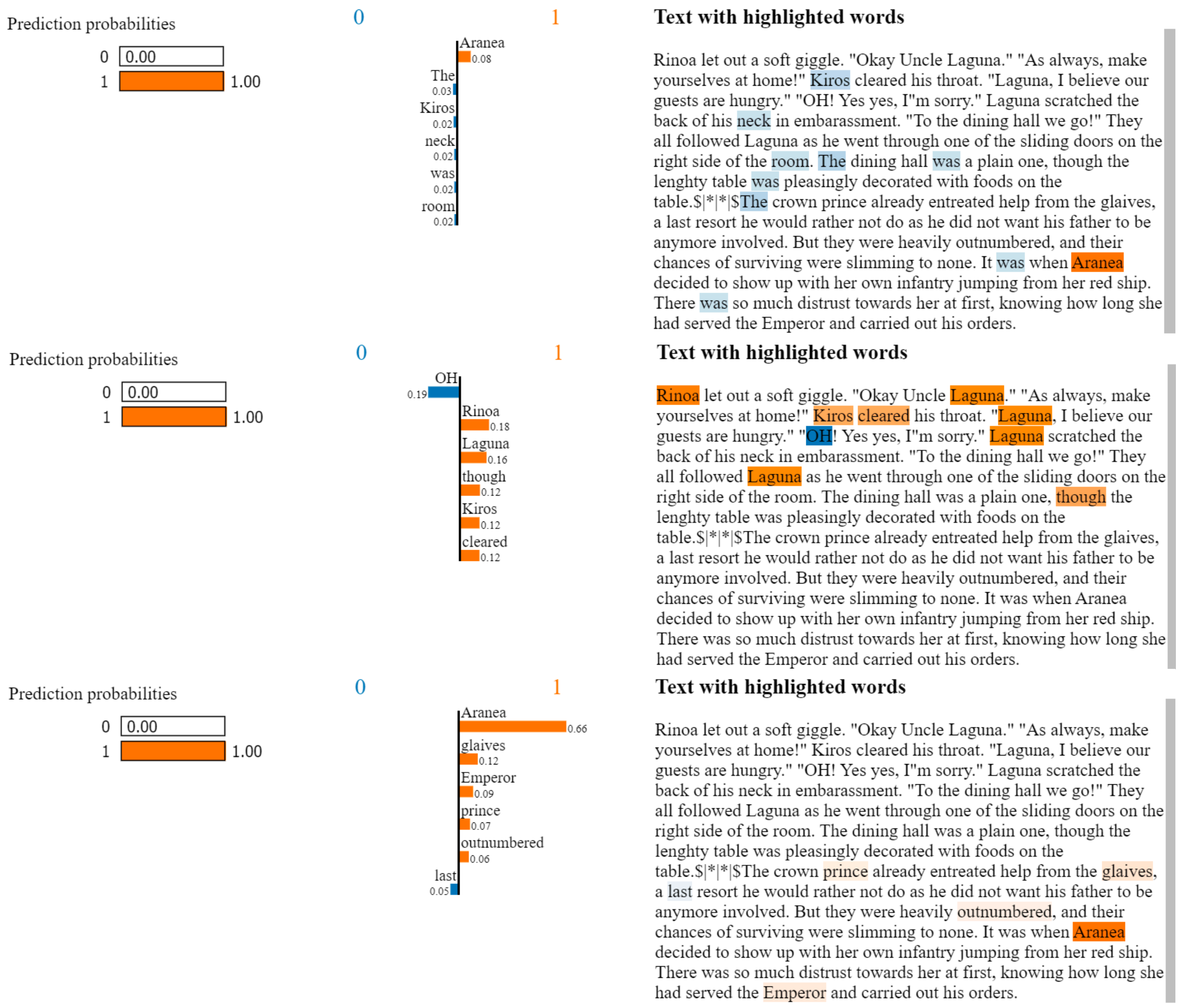
7.4 Feature extraction with pair-aware perturbations

In this section, we provide the analysis of features that were extracted for different segments. We generate LIME explanations for random segments of the selected pairs, assess the correctness of LIME estimations for extracted important features by removing them separately, and then check their joint importance upon simultaneous deletion. After that, we proceed with investigating the behavior of other potentially important features with and without the most important ones removed to get the initial understanding of the influence of stylistic changes on the model predictions.

For segments with predicted class 1, as mentioned above, basic LIME assigns comparatively low weights to the top-6 features in segments, though these weights are slightly higher than previously. LIME-pair, however, produces explanations with much higher weights for both the “left” (Figure 7.3, middle) and the “right” (Figure 7.3, bottom) modes.

For predicted class 0, LIME-pair, in turn, is unable to assign any weights to top features for since explaining texts for this class requires controlling dissimilarity rather than similarity. Basic LIME, on the contrary, can produce meaningful explanations with assigned weights. These weights, however, do not linearly approximate the probabilities or the logits but can still serve as means of qualitative estimation.

Therefore, the most illustrative results for our task can be achieved by using both types of LIME explainer: LIME-pair in “left” and “right” modes for class 1 and basic LIME for class 0.



Rinoa let out a soft giggle. "Okay Uncle Laguna." "As always, make yourselves at home!" Kiros cleared his throat. "Laguna, I believe our guests are hungry." "OH! Yes yes, I"m sorry." Laguna scratched the back of his neck in embarassment. "To the dining hall we go!" They all followed Laguna as he went through one of the sliding doors on the right side of the room. The dining hall was a plain one, though the lenghty table was pleasingly decorated with foods on the table.$|*|*|$The crown prince already entreated help from the glaives, a last resort he would rather not do as he did not want his father to be anymore involved. But they were heavily outnumbered, and their chances of surviving were slimming to none. It was when Aranea decided to show up with her own infantry jumping from her red ship. There was so much distrust towards her at first, knowing how long she had served the Emperor and carried out his orders.

Rinoa let out a soft giggle. "Okay Uncle Laguna." "As always, make yourselves at home!" Kiros cleared his throat. "Laguna, I believe our guests are hungry." "OH! Yes yes, I"m sorry." Laguna scratched the back of his neck in embarassment. "To the dining hall we go!" They all followed Laguna as he went through one of the sliding doors on the right side of the room. The dining hall was a plain one, though the lenghty table was pleasingly decorated with foods on the table.$|*|*|$The crown prince already entreated help from the glaives, a last resort he would rather not do as he did not want his father to be anymore involved. But they were heavily outnumbered, and their chances of surviving were slimming to none. It was when Aranea decided to show up with her own infantry jumping from her red ship. There was so much distrust towards her at first, knowing how long she had served the Emperor and carried out his orders.

Rinoa let out a soft giggle. "Okay Uncle Laguna." "As always, make yourselves at home!" Kiros cleared his throat. "Laguna, I believe our guests are hungry." "OH! Yes yes, I"m sorry." Laguna scratched the back of his neck in embarassment. "To the dining hall we go!" They all followed Laguna as he went through one of the sliding doors on the right side of the room. The dining hall was a plain one, though the lenghty table was pleasingly decorated with foods on the table.$|*|*|$The crown prince already entreated help from the glaives, a last resort he would rather not do as he did not want his father to be anymore involved. But they were heavily outnumbered, and their chances of surviving were slimming to none. It was when Aranea decided to show up with her own infantry jumping from her red ship. There was so much distrust towards her at first, knowing how long she had served the Emperor and carried out his orders.

Figure 7.3: *Segment-level LIME features, true class “1”, bag-of-word setting, for LIME-base (top), LIME-pair in “left” mode (middle), and LIME-pair in “right” mode (bottom)*

We note that, in principle, LIME-pair is also able to produce feature extraction with non-zero weights assigned even on the text level, but given the averaging procedure that may hide the importance of certain features we recommend generating segment-level explanations and counting the features extracted by them instead if needed.

7.4.1 Extracted features overview

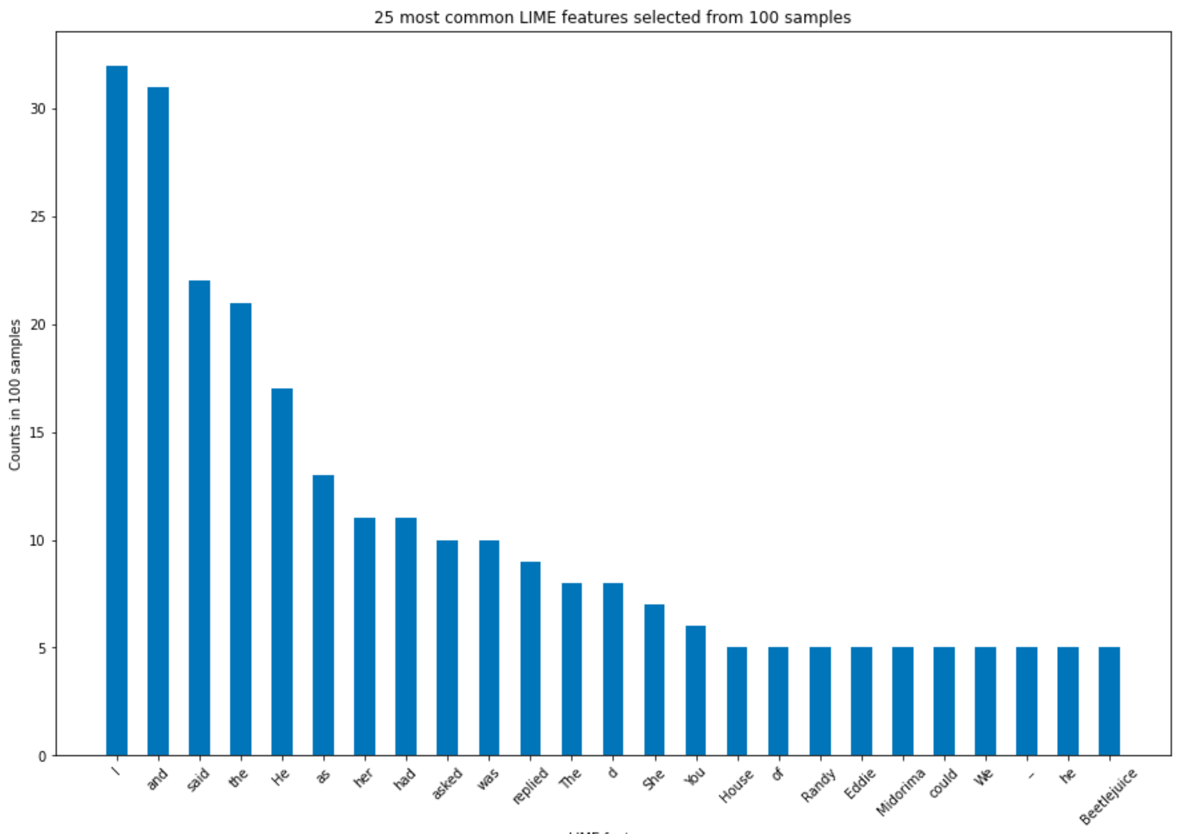


Figure 7.4: *counts of 25 most common features extracted from 100 sample segments*

Following this procedure, we obtained important features for 100 sample texts. 25 most frequent ones are displayed in Figure 7.4. We can observe that the most commonly selected are function words, including conjunctions (“and”, “as”), various personal pronouns (“I”, “He”, “her”, “You”, and more), auxiliary verbs in full and short (“d”) forms, reporting verbs (“asked”, “replied”), and some proper names that were found in more than one segment (for each text in the sample, 5 segments were considered). Finally, an underscore, which is apparently disregarded by the non-character filter used in LIME-text, is also extracted. Other types of punctuation are not present due to the filtering.

The majority of these features are words of very high frequency, which makes them suitable stylometric features: even slight differences in their frequency or in the patterns of usage can be indicative of the individual style. First- and third-person pronouns enable representation of the narrative perspective: whether the story is narrated from a first- or third-person view. Reporting verbs, in turn, are highly variable, and their diversity or repetitiveness can also be subject to individual style. Names and proper nouns, on the contrary, are typically involved in modeling the topic of the narrative, yet they may incorporate numerous other features: the author’s preferences in character selection and in choosing to use names instead of other means of reference (co-referring nouns, pronouns) can also be indicative of the individual stylistic fingerprint, and the involvement of multiple pronouns as important features supports this view.

7.4.2 Relations between LIME estimations and prediction logits

Neither the original explainer nor the augmented version assigns weights that correspond to the probability of the class prediction. However, it was discovered that in many cases the weights assigned by LIME-pair approximate the change in the model's logits after removing the corresponding feature.

For example, for the text displayed in Figure 7.5, the original prediction had class logits [-5.531711, 5.7830925]. Removing the word *Aranea* (with the weight assigned by LIME being 0.66) resulted in the new logits [-4.927722, 4.5944533], so that the difference for class 0 logits was 0.6). Removing the word *Rinoa* led to logits [-5.381693, 5.3939533] (observed difference 0.15, predicted weight 0.18). This pattern repeats for the word *Laguna* (predicted weight 0.16).

This pattern, nevertheless, cannot be observed for all top features, likely due to the non-linear and interconnected nature of some of these features. When a certain feature is removed together with another one at the premutation stage and together they create an important complex feature, it may increase the average weight of the feature in question among all permutations and lead to an incorrectly high linear approximation that doesn't correspond with the change observed upon the removal of solely this parameter from the input segment.

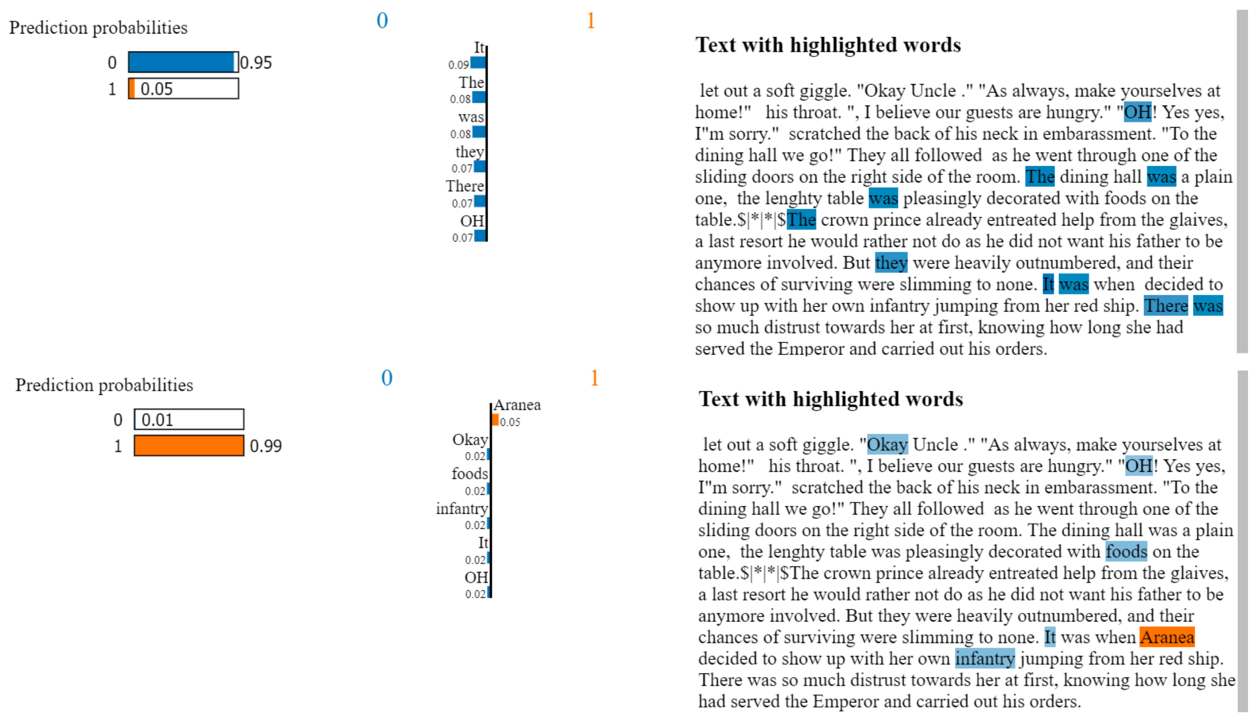
Thus, beyond the top 2-3 parameters, for which the weights approximate logits rather accurately, only qualitative estimation is possible — namely, that features with higher assigned weight contribute more to the logits, and features with positive weights contribute to class 1, and vice versa. The precise value of that contribution, though, depends on the co-occurrence with other features.

7.4.3 Co-occurrence of important features

Let us discuss these interconnections in more detail. Comparison of logits obtained after removing one or more important features extracted by LIME and other pre-selected candidate features in different combinations shows that even a local linear approximation of the model's behavior is implausible since the contribution of any feature is not constant and depends on its position, context, the total number of occurrences, and the presence of other features. Furthermore, these dependencies are by no means limited by the word level: the prediction can be influenced by the punctuation and stylistic variations thereof, by casing, and spelling mistakes.

First of all, it appears that the most important features extracted by LIME, which indeed were shown to contribute to the corresponding class, tend to accumulate their value. For example, removing the word *Aranea* increases the logit of the incorrect class 0 by roughly 0.6, which is insufficient to change the prediction. However, when the other five features contributing most strongly to class 1 (*Rinoa*, *Laguna*, *though*, *Kiros*, *cleared*) are also removed, so that the resulting class logits are [-2.8962233, 1.7099165], the contribution of removing Aranea to class 0 increases to 4.34, which results in the change of the predicted label logits: [1.4439765, -1.4362624]. This change if shown in Figure 7.5. Thus, the influence of top features is the strongest when no other top features occur in the text, and it drops substantially when the other features indicative of the same class are already present.

This claim can be generalized to other features, which are note highlighted by LIME, as well: they tend to show a much stronger influence on the prediction when the most important features, especially names, are removed from the text. Such behavior seems logical: without the names that could have been learned to indicate certain authorship, the model can no longer rely on such shortcuts and needs to make predictions based on other features of the text, with naturally lower confidence and robustness. Figure 7.5 (top) shows that, with the absence of 6 features contributing to class 1, the remaining features point at class 0, and these are mainly pronouns and functional words, which are essentially traditional stylometric parameters.



Text with highlighted words

let out a soft giggle. "Okay Uncle ." "As always, make yourselves at home!" his throat. ", I believe our guests are hungry." "OH! Yes yes, I"m sorry." scratched the back of his neck in embarassment. "To the dining hall we go!" They all followed as he went through one of the sliding doors on the right side of the room. The dining hall was a plain one, the lenghty table was pleasingly decorated with foods on the table.$|*|*|$The crown prince already entreated help from the glaives, a last resort he would rather not do as he did not want his father to be anymore involved. But they were heavily outnumbered, and their chances of surviving were slimming to none. It was when decided to show up with her own infantry jumping from her red ship. There was so much distrust towards her at first, knowing how long she had served the Emperor and carried out his orders.

Text with highlighted words

let out a soft giggle. "Okay Uncle ." "As always, make yourselves at home!" his throat. ", I believe our guests are hungry." "OH! Yes yes, I"m sorry." scratched the back of his neck in embarassment. "To the dining hall we go!" They all followed as he went through one of the sliding doors on the right side of the room. The dining hall was a plain one, the lenghty table was pleasingly decorated with foods on the table.$|*|*|$The crown prince already entreated help from the glaives, a last resort he would rather not do as he did not want his father to be anymore involved. But they were heavily outnumbered, and their chances of surviving were slimming to none. It was when Aranea decided to show up with her own infantry jumping from her red ship. There was so much distrust towards her at first, knowing how long she had served the Emperor and carried out his orders.

Figure 7.5: *Segment-level LIME features, true class "1", bag-of-word setting, for LIME-base. In the top example, 6 features previously identified as important for class "1" are removed (Rinoa, Laguna, though, Kiros, cleared, Aranea); see Figure 7.3. In the bottom example, one of them (Aranea) is brought back*

Such influence was not only discovered for co-occurrences of the word features. Some other cases are presented in Table 7.1. For example, the opening quote before *OH* (*OH* itself is marked as a #1 feature for class 0, as seen in Figure 7.3 (middle)) turns out to be important for class 1. Although its removal is unimportant when other features are present (it changes logit of class 0 by 0.04), it reaches 2.753 in case the 5 important features for class 1 (*Rinoa*, *Laguna*, *though*, *Kiros*, *cleared*) are removed beforehand, and results in the prediction label change: [-0.14309846, -0.2922034]. That opening quote does not show a strong correlation with any of the top features in particular, so it is the absence of all 5 that matters.

Importantly, quotes are only contained in the first part of the segment, so it might seem that removing any of them should make the texts more similar. However, the model's behavior indicates quite the opposite result, and removing all quotes when top features are removed results in an even higher increase of class logits: [2.9483943, -3.1187959]. This pattern can only be observed when no top features are present. Otherwise, the increase is marginal: [-5.3608084, 5.3494406].

Example feature	Influence on class 0 logits upon removal	
	Top-5 features present	Top-5 features absent
"Aranea"	0.604	4.340
Singe quote before "OH"	0.040	2.753
All quotes	0.171	5.844

Table 7.1: *Influence of removing selected features on prediction logits with and without top-5 features removed, for an example text from Figure 7.5*

7.4.4 Adversarial examples

We proceeded with further experiments by introducing adversarial examples that involved punctuation and typographic variations, as well as removal and replacement of various features. In general, we observed that when the top features were present, no significant changes could be observed. After removing them, alterations became a lot more salient. Results of different alternations are shown in Table 7.2

Typographic variations

Firstly, we checked the typographic variations. Replacing double quotes (“) with single quotes (‘) did not result in a change of the predicted label but decreased the overall similarity between texts by pushing class 0 logits higher and class 1 — lower: [-1.8707004, 0.9313528].

After that, we considered available stylistic variations. The first text contained 3 exclamation marks that could be replaced with full stops without significant change in the meaning and were therefore subject to the author’s stylistic choice.

Removing and replacing punctuation

Changing all full stops into exclamation marks dramatically increased the similarity: [-4.64954, 4.0420275]. Making an identical change in only one of the texts did not make such a big difference until we also changed all exclamation marks to full stops in the other one: [2.222923, -2.2992437]. Simply replacing all exclamation marks with full stops did not yield any significant results but slightly increased the probability of class 0 (which, again, might seem counter-intuitive given that such change makes the texts more similar).

When we replaced exclamation marks with question marks, which produces a meaningful semantic change, the decrease in similarity was a bit higher [-2.616405, 1.396568], yet still without changing the predicted label.

Removing punctuation significantly increased the similarity of the texts (i.e., the logit of class 1): removing all punctuation resulted in logits being pushed back to [-5.441371, 5.5717177], removing solely full stops — in [-4.8784447, 4.341698], while removing solely commas — in [-5.252601 5.1758327], which emphasizes the importance of different uses of commas for the difference between the styles of various authors.

Thus, we could increase the similarity of the texts by removing punctuation or making uniform replacements in both texts. Increasing the dissimilarity turned out to be a harder problem: so far, we could do it only by making directly opposite replacements of “.” and “!”. Other cases in which the alteration resulted in a different prediction are adding more full stops to the first text (to the places where there were no punctuation marks) while also removing full stops from the second text where possible or replacing them with stylistic alternatives (commas, semicolons, double dashes): [1.8393167, -1.9136158].

Example punctuation feature	Class 0 (“different-author”) logits	Class 1 (“same-author”) logits	Interpretation
Default, top-5 features removed	-2.8962233	1.7099165	Texts are similar
Double quotes replaced with single	-1.8707004	0.9313528	Similarity decreased
Full stops replaced with exclamation marks	-4.64954	4.0420275	Similarity increased

Text one: exclamation marks to full stops; text two: full stops to exclamation marks	**2.222923**	**-2.2992437**	Similarity decreased; texts are dissimilar
Exclamation marks replaced with question marks	-2.616405	1.396568	Similarity decreased
Removing all punctuation	-5.441371	5.5717177	Similarity increased
Removing full stops	-4.8784447	4.341698	Similarity increased
Removing commas	-5.252601	5.1758327	Similarity increased
Text one: more full stops added; text two: full stops replaced with alternatives	**1.8393167**	**-1.9136158**	Similarity decreased; texts are dissimilar
Adding nonessential correctly placed commas	**1.5631607**	**-1.6246212**	Similarity decreased; texts are dissimilar
Adding non-grammatical commas	-2.0973256	1.0720851	Similarity decreased

Table 7.2: *Influence of input ablation and alteration on prediction logits for an example text from Figure 7.5*

Adding new punctuation

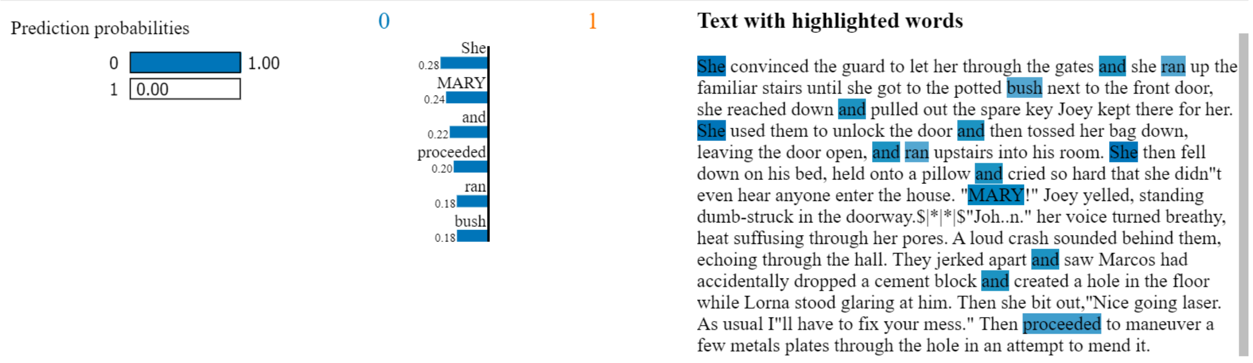
We then considered adversarial examples that added new punctuation marks to the positions where no other symbols were present. It turned out that new instances of commas have a stronger influence than those obtained by replacing other punctuation.

Adding new commas to the first text yielded different results depending on their syntactic role and co-occurrence. When we added several unnecessary commas to both texts, isolating *I believe* as a parenthetical clause resulted in a significant change and a different label: [1.5631607, -1.6246212]. Adding a non-grammatical comma (for example, after *let out*) didn't have a similarly strong influence: [-2.0973256, 1.0720851].

We can conclude that it is possible to control the similarity by making changes to punctuation marks in certain locations, particularly where they are optional or believed to be so or can be replaced by alternatives. Although the quantity of changed symbols does influence the intensity of changes, sometimes just one symbol is crucial for making the correct prediction, contributing almost 70% of the total weight of all such symbols. However, these relations are complex and non-linear: adding more commas to random places of one text does not necessarily make it significantly less similar to another one and combining two features with positive influence may result in a negative influence.

The non-linearity can be explained by the fact that BERT can encode contextual and syntactic information, and therefore commas between the components of a complex sentence or after a parenthetical clause are treated differently. Besides, commas can be strictly grammatical, or subject to stylistic variation, or explicitly non-grammatical.

Controlling capitalization



Text with highlighted words

She convinced the guard to let her through the gates and she ran up the familiar stairs until she got to the potted bush next to the front door, she reached down and pulled out the spare key Joey kept there for her. She used them to unlock the door and then tossed her bag down, leaving the door open, and ran upstairs into his room. She then fell down on his bed, held onto a pillow and cried so hard that she didn"t even hear anyone enter the house. "MARY!" Joey yelled, standing dumb-struck in the doorway.$|*|*|$"Joh..n." her voice turned breathy, heat suffusing through her pores. A loud crash sounded behind them, echoing through the hall. They jerked apart and saw Marcos had accidentally dropped a cement block and created a hole in the floor while Lorna stood glaring at him. Then she bit out,"Nice going laser. As usual I"ll have to fix your mess." Then proceeded to maneuver a few metals plates through the hole in an attempt to mend it.

Figure 7.6: *Segment-level LIME features, true class "0", bag-of-word setting, for LIME-base.*

For another text segment, shown in Figure 7.6, the top 6 most important features were: *She*, *MARY*, *and*, *proceeded*, *ran*, *bush*. Of these, *MARY* is of significant interest. We considered removing that word completely, which slightly increased the probability of class 0. After that, we replaced that name with another capitalized name of the same length (*JOHN*), and the probability increased even further. Interestingly, replacing capitalized *MARY* with not capitalized *Mary* resulted in an even higher increase. Thus, the presence of capitalization turns out to be even more important than a particular name. This suggests that casing is indeed a meaningful feature for this model.

The difference between uppercased and lowercased instances of the same word can be more salient than that between different words written using the same case. This goes in line with the stylometric theory: typographic differences that are controlled by the author are often optional and can therefore indicate individual stylistic preferences. Those not fully controlled by the author, such as fonts and margins, need to be used with caution since they may have been changed by the published or during the preprocessing. In fact, words can be lowercased at the preprocessing stage as well, but we preserve the original casing as it was in the dataset.

Such importance is also explainable from the BERT perspective: in the regular casing, most conventional names (*Mary*, *Joshua*, *Alexander*) and some unconventional (*Laguna*) are processed into a single token. Others are treated by subword units (*R* + *##ino* + *##a*). When the name (or any other word) is unexpectedly capitalized, it is segmented into a list of subword tokens with one or more capital letters. Thus, identical words with and without uppercasing are represented as different sets of tokens, so assigning different weights to them is justifiable.

7.4.5 Feature importance discussion

The nature of internal connections between the features observed and analyzed above is diverse. For example, an opening quote before *OH!* is clearly only important when *OH!* itself is present, but the exact rule behind that observation can have various explanations: for example, that it is typical for this author to start a direct speech with capitalized exclamations, or that the author generally uses quotation marks to convey direct speech, and closing and opening quotes often follow one another, so if one of them is missing in the middle of the direct speech narrative, it is an argument against that author's authorship.

It is also important to re-iterate that these features show up when the most salient and constant features, which are predominantly names in the examined texts, are removed. Otherwise, changes in other features only show a moderate influence on the final prediction, normally without changes in the label.

Combining observations from multiple texts, we can conclude that segments seem to have different degrees of robustness, and this robustness does not necessarily correspond with confidence. This consideration makes sense given the non-linear nature of many features: in some positions or in certain combinations, the features can have an extremely strong influence on the output logits.

Besides, even with confidence being 1, the actual logits may differ substantially across segments: a prediction with logits around [-7, 7] is expected to be more robust than that with [-4, 4], even though the probabilities will be identical after the Softmax activation.

Such differences can be expected from the model with such architecture as ours since the logits are averaged across all 30 segments of a given text. Therefore, if the model is particularly certain about a segment, it can surely assign more extreme values to the logits to move the averaged prediction further towards the correct prediction. The segment about which the model is not that certain can just barely pass the threshold or even remain incorrect, which is not critical if their logits for both classes are close to 0.

In general, we can conclude that the features extracted by LIME can in principle be interpreted as stylometric and mostly belong to the category of word-level features related to function word counts. The joint usage of the two versions (standard LIME and LIME-pair) enables generally reliable estimation of weights at least for the most salient features, thus proving that the problem of original LIME was likely in the inability of usual perturbation to control similarity. Beyond the most salient features, however, feature importance can still be estimated qualitatively in terms of relative contribution and importance for a particular class.

8. Attention analysis

This chapter covers the analysis of the attention in the fine-tuned BERT model that can be relevant for classifying texts in authorship verification. We then proceed with suggesting an algorithm that takes into account the weights of attention emitted from CLS tokens and selects the tokens to which CLS attends most commonly or with the highest intensity. This algorithm is used for feature extraction. The extracted features are then analyzed and compared to the features highlighted by the LIME explainer, the features suggested as important by recent works in the field of authorship attribution, and the features traditionally used in authorship attribution.

8.1 CLS attention overview

We started with analyzing and counting the weights in attention matrices in all layers and heads in each example from the sample of 100 text pairs. For each layer separately, we considered attention from CLS tokens. For each head, we took 10 tokens with the highest attention outcoming from the CLS token. Then, we combined the lists of these tokens from all heads and counted their occurrences. The result can be seen in the Appendix A.

The resulting data provided information about the rank and frequency of CLS attention to each token for a *given text pair* at each layer, thus showing whether it was the one universally attended by CLS in all heads or selected only by some heads. Using both rank and counts enabled us to track the involvement of the SEP token throughout the layers and discover the layers at which the largest part of attention from CLS is poured into that token. Besides, we could see specific tokens that are frequently attended in particular texts but are not overall common due to their uniqueness, such as particular names or their parts.

Additionally, we counted how many times these tokens were selected in *all* examples from our sample in each layer. In this setup, we counted the occurrence of a token in a particular instance's list of frequently attended tokens only once to mitigate the difference in frequencies within one text and avoid assigning high frequencies to tokens that are attended often yet only in one particular instance.

This observation provided more general information about the distribution of frequently attended tokens over all instances. Given that the instances and the number of heads remain unchanged throughout the layers, changes in that distribution enabled us to observe which tokens are more commonly attended by CLS at some particular layers and whether some general trends can be retrieved.

8.1.1 Preliminary per-layer analysis of CLS attention

A certain pattern could be noticed for the behavior of the SEP token. SEP token is claimed to be a "dustbin" for attention (Clark et al., 2019) in case nothing meaningful for that combination of layer, head, and token was found in the sentence. It was consistently among the most popular ones. However, its rank and overall counts (limited to 24 per layer: 2 in the text, to be attended by each of 12 heads) differ substantially across levels.

At the first layer, frequencies are inconsistent, but SEP typically appears 7–11 times, ranking between 1 and 7. In general, the scope of attention of CLS at this layer seems diverse (2119 different tokens), but the most common are: "the", "a", "and", "to", "of", punctuation (full stops, commas, question and exclamation marks, quotes, hyphens, semicolons), some subword tokens: frequent affixes ("##es", "##ed", "##ly", "##ing") and frequent name constituents (mostly one- or two-character tokens) — tokens that are generally common in texts, but also meaningful for style classification. They are followed by a large number of tokens attended by CLS frequently only in a fraction of segments, mostly segments from one text. These involve more specific name constituents ("##us" for Rome-themed texts, "##mir" for the Lord of the Rings-themed texts), some single-token names, pronouns, and frequent verbs.

The second layer is less diverse (1526 distinct tokens) and is concentrated around a small number of frequently attended tokens: SEP, CLS, and "." appear in all examples, but apart from them only a few hyper-frequent words appear in multiple examples ("the", "The", "a", "was", "I"). Since the texts in the sample are literary works, it is no surprise that the reporting verb in the past tense (such as "asked" and "said") are also among such frequent words. Importantly, the rank of tokens is very consistent: "." is the most frequent in almost all examples, almost universally followed by SEP and CLS. The number of SEP tokens being attended is also much higher, varying around 19–24 per instance.

This pattern repeats in Layer 3, with the difference being in much higher attention for quotes (94%) and the presence of other punctuation. Layer 4 adds commas to most common tokens (also 94% compared to 69% at layer 3) and pays more attention to frequent verbs and pronouns ("said" in 39% vs 10% at layer 2).

At layers 7–8 question mark finally appears among the most frequent tokens (41% at layer 8, 24% at layer 7 vs 9% at layer 6) and shows even more attention to narrational verbs "said", "asked", and "replied" (43%, 32%, and 14% respectively). Pronouns, most notably "I", gradually become more important (for "I": 38% at layer 7, compared with 9% at layer 2).

At layer 9, the rank of SEP drops. It was most universally ranked #1 at layer 7 and was declining since then. At the final layer, SEP if almost never the most frequent token. The most frequently attended ones are full stops, commas, and quotes, followed by all forms of personal pronouns, narrational verbs and adverbs, relative pronouns, conjunctions, and subword tokens.

8.1.2 Discussion of the general CLS behavior

Combining the observations above, we can conclude that CLS attends more, and also more diversely, at the first three layers, and after that the continuously growing share of its attention is given to the SEP tokens, maximizing at layer 7. Then the importance of SEP starts declining but drops most substantially at the very last layer. This can be explained by the fact that the intermediate layers are often concerned with syntactic analysis of various kind, and fewer processes in these layers are related to shaping the CLS representation.

The intermediate layers show different preferences for some specific tokens: some favor a particular punctuation mark or a particular part of speech. The features that are overwhelmingly frequently attended as important are three basic punctuation marks (full stops, quotes, and commas). They are followed at some distance and with layer-wise variation by pronouns, conjunctions, prepositions, and some reporting verbs that are frequently used in storytelling (aside from those mentioned above, these are also “thought”, “answered”, and “stated”).

For the names and name constituents, it is harder to track the exact rank and frequency changes due to their uniqueness. It can be nonetheless observed that they repeatedly arise among the most frequent in higher layers but only truly shine at the final layer, where they often turn out to be the most important ones.

8.2 Attention visualization overview

Although the general analysis of CLS attention in all layers provided high-level statistics of the importance of different types of tokens and the amount of potentially irrelevant attention (the one directed to SEP), it did not allow for proper exploration of the attention directed at particular words and punctuation. Besides, the analysis of raw matrices did not provide information about the scope of attention in each head and its specificity.

Moreover, such approach only enabled us to consider attention outcoming from CLS, whereas the types of attention that can be relevant for shaping the CLS representation may also include attention directed at CLS from tokens frequently attended by CLS and attention between tokens that have already been marked as relevant to CLS or between such relevant tokens and some other categories. Such variety is motivated by the fact that when CLS attends to a particular token, it gets access not only to the token itself but to its embedding which includes a representation of different information about the token and its context.

For this reason, we proceeded with using a visualization tool, namely BertViz (Vig, 2019), which is an interactive tool that enables visualization of Transformer attention. It can be run in interactive Python notebooks and has native support of the HuggingFace Transformers framework, including both the model and the tokenizer.

The attention maps used for the analysis of BERT syntactic capabilities in (Jawahar et al., 2019) was limited to the maximal length of 128 tokens. Thus, the segments examined by them are twice as short as the ones in our base model. The computational cost of processing the maps is considerably higher in our case, and the maps for different layers can only be visualized separately, without the option of choosing the layer on a single visualization, which is usually possible with BertViz (Vig, 2019).

This tool is theoretically able to create a general visualization for a given input, allowing the user to choose the layer, the head, and the scope of attention. However, in our case, BertViz was unable to process all layers for the text pair, likely given the large size of the input. To overcome this limitation, we started with averaging the attention in each head over all layers and proceeded with the analysis of all heads in all layers separately. We analyzed the resulting attention maps and compared them with similar maps produced by an untuned BERT model for text classification to see whether the observed relations are general properties of BERT embeddings construction or they are specific to the task of authorship verification.

8.2.1 Averaged attention visualization

Averaging attention of one head across layers provided general insights into the difference between the fine-tuned and the untuned models. We observed that the basic patterns, such as attention to next token, previous token, and same token, are typically preserved in both models. Semantic relations, such as links between words from the same semantic group related to the same topic, is also typically retrievable from both models, though they can be stronger in the fine-tuned one. Other relations are typically dissimilar. In the untuned model, a lot more attention is given to full stops, and such attention is a lot less selective: it may be coming from all words in the input pair uniformly. A lot more common are also various types of attention that select syntactic constructions. There are also significantly fewer long-distance relations: most types of attention in the untuned model either have a fixed local window (for example, 5–10 tokens or the scope of one sentence) or are spread uniformly. All these relations, in principle, go in line with the observations of Clark et al. (2019). The fine-tuned model selects fewer types of constructions but with higher emphasis on them and often attends to similar constructions on different layers. It also utilizes a large number of long-distance relationships, usually to link the tokens from a certain category (such as reporting verbs or special characters) regardless of their location.

The specific patterns observable in averaged attention maps include diverse attention involving subword tokens: some link all tokens from one word between each other; in others, all subword tokens attend to one, typically the initial token.

We also observed repeating attention that involves quotes. The most common one is the attention to the quotation mark from the segment preceding the next citation. Interestingly, closing quotes are consistently more important than opening ones (both examples are for closing). Opening quotes are of least importance when they directly follow a full stop. Despite the identical symbols being used for quotes and apostrophes in the preprocessed data, their ambiguity seems to be resolved: apostrophes that can be mistaken for a closing quote do not receive much attention. In Figure 8.1, the apostrophe only gets attention from “ve”, although it stands in the position making it a potential closing quote and should attend the upcoming words. Besides, some quotation marks are attended from other quotation marks, even distant ones from another text, which suggests that counting might be going on in some levels.

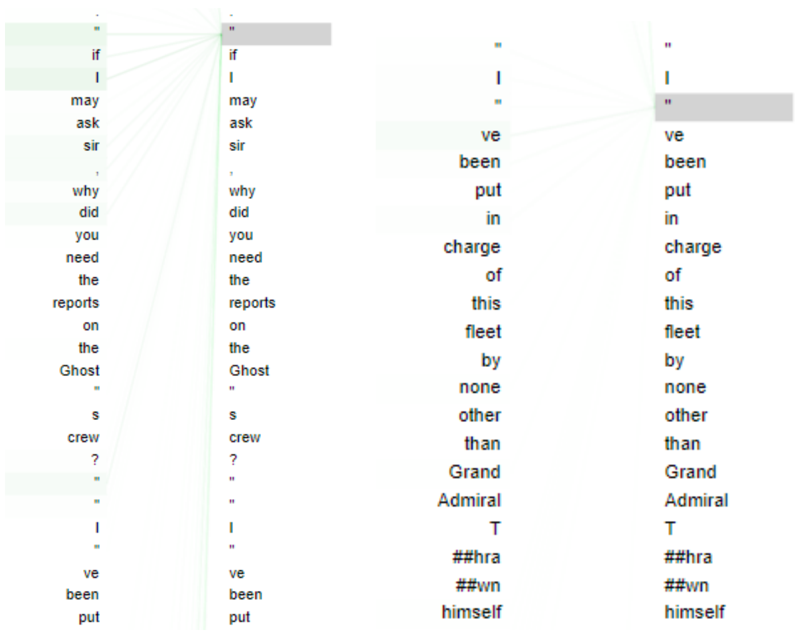


Figure 8.1: *Attention received by a quote and an apostrophe (Head 3, averaged across layers)*

Another specific attention involves reporting verbs (that is, verbs used for retelling, quoting, describing and paraphrasing a discourse). Noticeable are long-distance relations between identical verbs ("replied" attending to all its instances) in both texts, as well as attention to other synonymous reporting verbs in another text ("stated" and "said" attending to "replied" in the other text, but not in the same one). Such pattern is absent in corresponding heads of the untuned model.

8.2.2 Per-layer attention visualization

These general considerations with averaging the values across layers do not allow for tracking relations for a particular layer. Given that the role of attention on different levels may vary significantly even for the same head, many types of attention may be concealed because of that averaging. Therefore, we proceeded with per-layer analysis of attention maps for each head.

We provide the table that summarizes the most relevant types of attention in Appendix A. This table includes the direction of the link, types and categories of tokens involved, strength, scope, and coverage of the relation. A more detailed description of these types, supplemented with illustrations and a brief discussion of the exceptions and specificities, is available online[1].

Here, we outline some of the most notable relations.

Head 5 in layer 3 features diverse relations for punctuation, shown in Figure 8.2. **Full stops** mostly receive attention from words within one sentence, including all direct speech inserted in it. Sometimes the scope is limited to the nearest preceding punctuation, which can be a comma or a quote. **Commas** may attend to the context from both sides but typically only get attention from preceding words. For the **quotes**, the closing one is consistently more popular: it is attended by tokens from within the direct speech utterances. Specific behavior is displayed by more rare punctuation symbol**s**: question marks are strongly attended by the subject, the predicate, the object,

[1] https://elite-shame-5a4.notion.site/Per-layer-BertViz-Analysis-for-AA-5689fa1720bf4cdf9f63827faf7ed235

and, if present, the question word of the interrogative sentence, though such selectivity is not typical for other sentence separators.

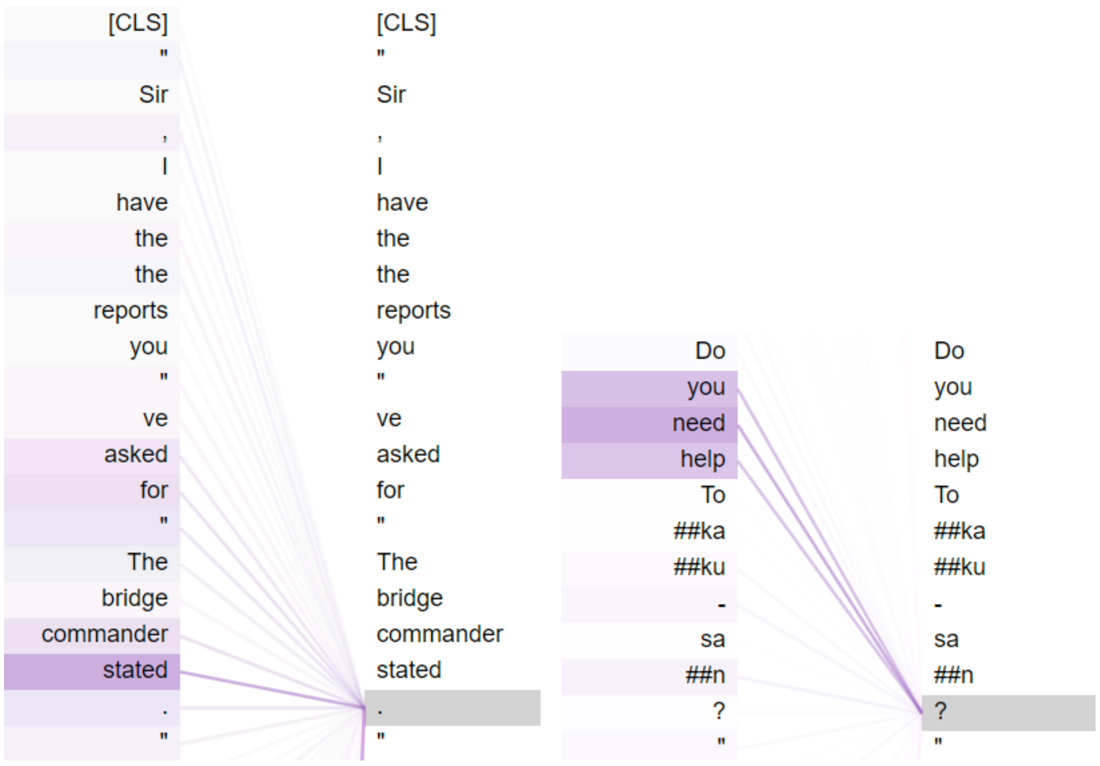


Figure 8.2: *Attention to punctuation, specific to its type (Layer 3, Head 5)*

Head 6 in layer 3 (Figure 8.3) shows more **remote identity relations** between words and symbols. Such attention is either concentrated on one identical token, not necessarily the nearest one, be it ahead or behind, or distributed among identical tokens. Interestingly, different forms of pronouns, such as “I” and “me” and “she” and “her”, are also linked. Strongest attention is between identical subword tokens, prepositions, functional verbs, wh-words, pronouns, and **wrongly spelled words**. Such a type of attention is not present in the untuned model.

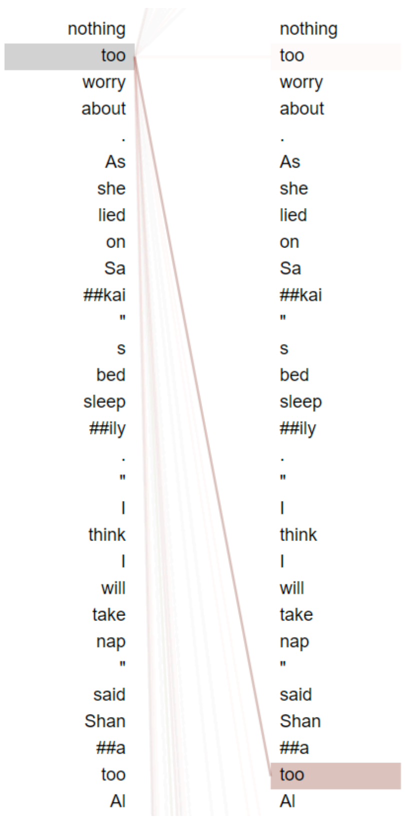


Figure 8.3: *Attention between identical important tokens, but not to itself. In this case, between two mistakes (words with redundant characters) (Layer 3, Head 6)*

Head 1 in Layer 4 (Figure 8.4) seems to be linking reporting verbs with quotations, as well as different names within a long distance from both texts (not with the same name).

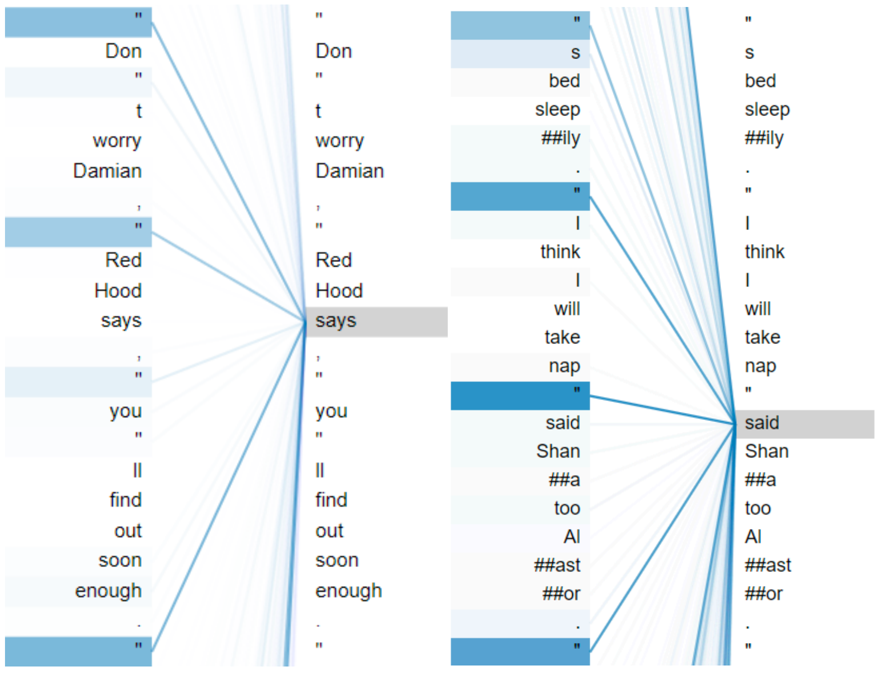


Figure 8.4: *Attention between reporting verbs and quotes (Layer 4, Head 1)*

In Head 2 of Layer 4, as well as in multiple other heads (see Appendix), we observe that CLS token attends a wide yet formalizable range of words, to which we will be referring as “important” words.

They include:

- dialect variations
- names
- reporting verbs
- some content words with stylistic variation, which can be replaced with a well-known synonym without substantial change in meaning (“fellows”, “kids”, “kind” as noun, “manner” meaning “way”, “acknowledge”, “becoming”), including many adjectives and adverbs (“entire”, “cute”, “slightly”, “kindly”, “sole”, “lately”, “presumably”, “just” meaning “only”, “although”)
- various function words with stylistic variation (“towards”, “as”, “once” as conjunction)
- optional full or short forms of function words, mostly auxiliary verbs (“will” and “ll”, “are” and “re”, “have” and “ve”), as well as variable verb forms, such as “gotten”
- specific vocabulary not related to the theme (“feminine”), infrequent words not related directly to the theme, including those processed into subword tokens (“furrow”**,** “trundle”, "mischievous”)
- mistakes (“too” instead of “to”, missing determiners (“take nap”)
- some epithets (“beet red”)
- rude and obscene words
- numbers (“15” instead of verbalization)

Note that this list only incorporates important words, excluding the punctuation and subword tokens.

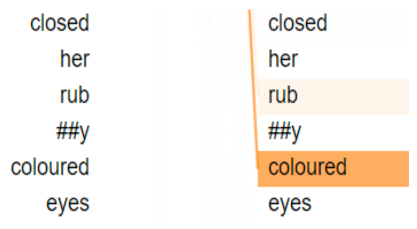


Figure 8.5: *Attention to punctuation, specific to its type (Layer 4, Head 2)*

Attention to dialect variations can be found in Head 2 of Layer 4 (Figure 8.5). Head 5 of Layer 5, shown in figure 8.6, demonstrates an interesting type of attention between full and short forms, but not from one full form to another. There also is attention between different forms of pronouns (“he” → “his”), from pronouns to neighboring names, and between names.

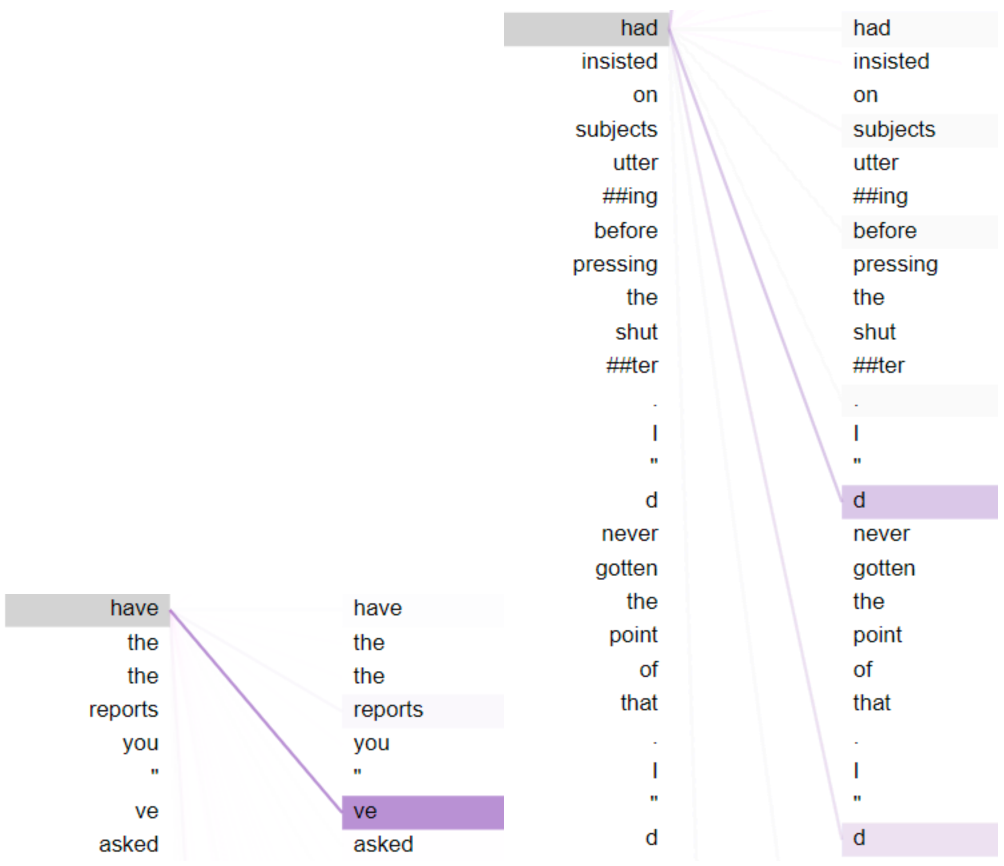


Figure 8.6: *Attention between full and short forms (Layer 5, Head 5)*

Head 6 in layer 6 displays attention between function words and punctuation. Attention is selective: it includes identity relation with variable weights, attention to different forms of a pronoun, or to other words of the same part of speech (such as auxiliary verbs). Outcoming attention is mostly headed forwards. This type of attention is also absent in the untuned model.

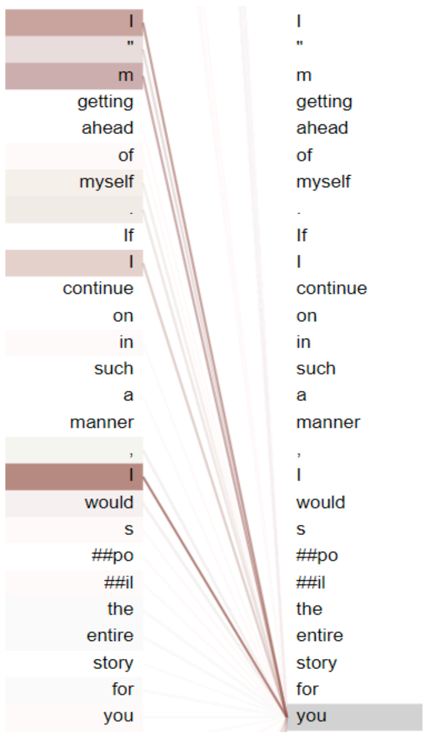


Figure 8.7: *Attention between pronouns (Layer 6, Head 6)*

Head 7 in layer 7 (Figure 8.8) has very selective attention to functional words and quotes to select subordinate clauses and appositive phrases. Conjunctions “if”, “although”, “as” are attended by the comma that isolates the conditional clause, and by the verb “would”, wh-words, or plainly the token following that comma. Importantly, “as” as a preposition is not confused with a conjunction, therefore part-of-speech ambiguity seems to be resolved for such relations. There also is a link between punctuation used to introduce direct speech and the verbs that describe that speech. Besides, in explanatory phrases isolated with commas, there is a backward attention from the closing to the opening comma. All these relations are absent in the untuned model.

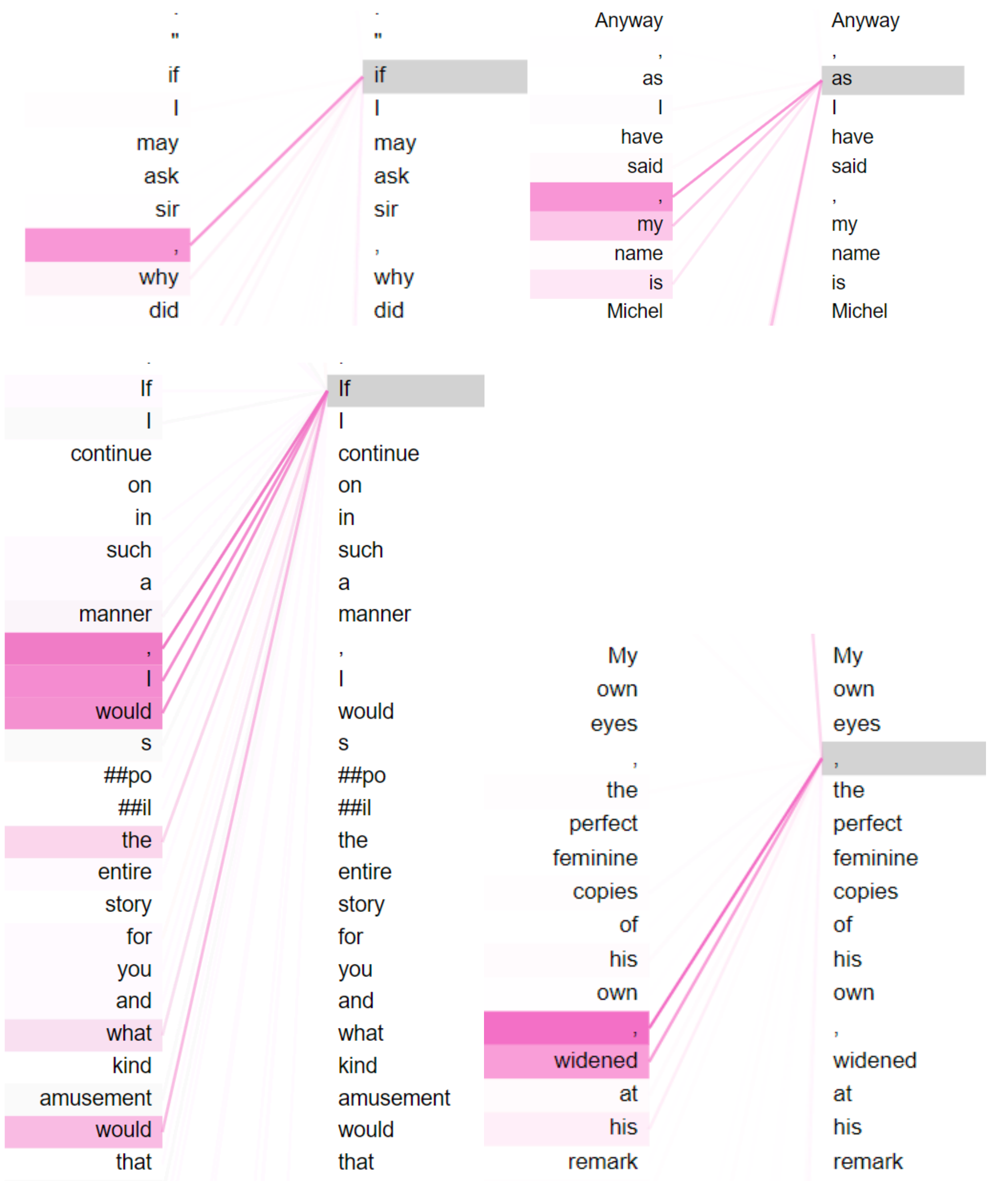

. " if I may ask sir , why did
. " if I may ask sir , why did
Anyway , as I have said , my name is Michel
Anyway , as I have said , my name is Michel
. If I continue on in such a manner , I would s ##po ##il the entire story for you and what kind amusement would that
If I continue on in such a manner , I would s ##po ##il the entire story for you and what kind amusement would that
My own eyes , the perfect feminine copies of his own , widened at his remark
My own eyes , the perfect feminine copies of his own , widened at his remark

Figure 8.8: *Attention selecting subordinate clauses and appositive phrases (Layer 7, Head 7)*

Head 11 in Layer 7, exemplified in Figure 8.9, marks such relations as that of determiner and noun, pronoun and adjective in such constructions as “my own”, and specific constructions, such as words joined with hyphens (”Boy-Who-Lived”). These types are in general the only ones in this head

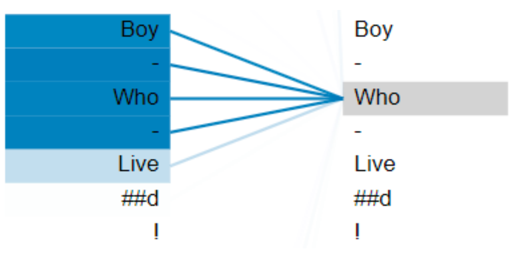


Figure 8.9: *Attention selects compound words with hyphens (Layer 7, Head 11)*

Head 3 of Layer 8 (Figure 8.10) provides a link from the reporting verb to the closing quote of the direct speech or to the relative pronoun used to describe indirect speech.

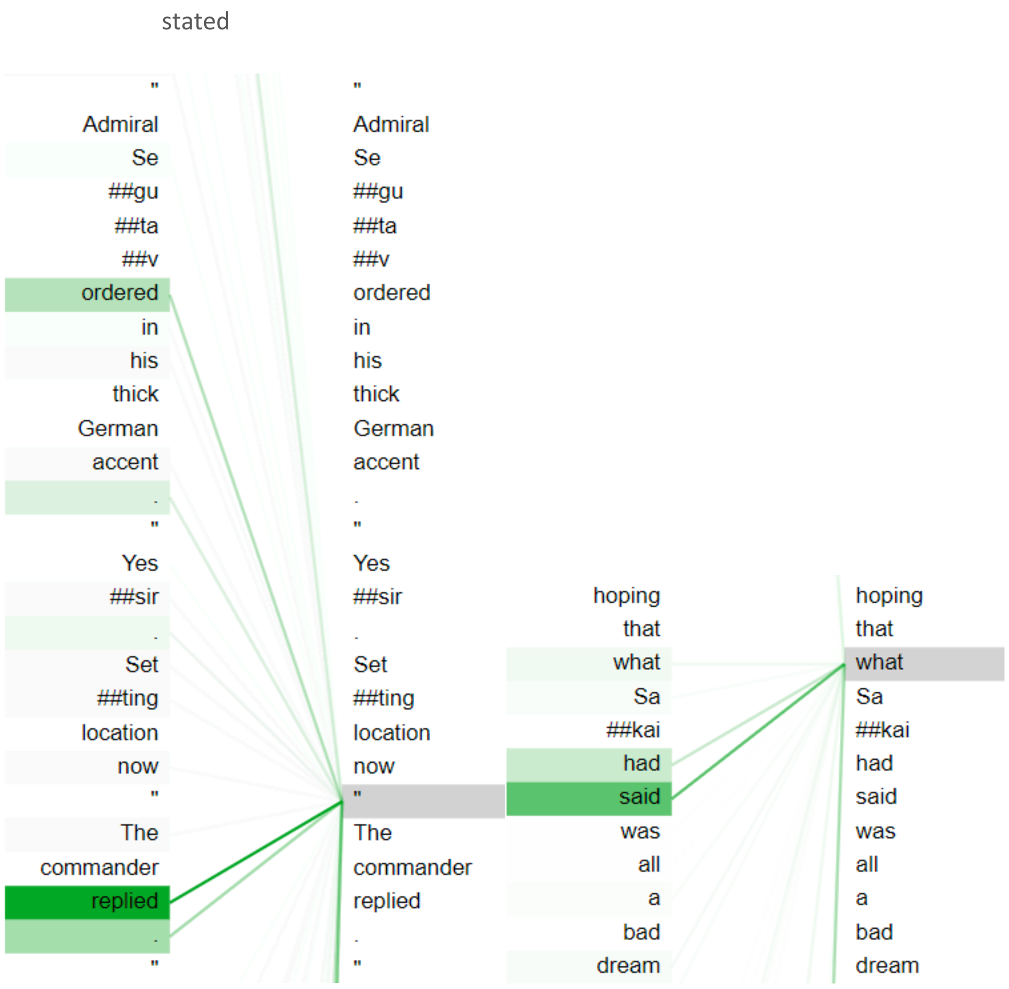


Figure 8.10: *Attention maps reporting verbs to the closing quote or relative pronoun*

In the later layers, CLS begins to show a lot more attention to punctuation, which did not happen widely on the previous ones. Along with attending to all types of punctuation, multiple heads in layers 10–12 feature attention from CLS solely to all full stops or quotes. In the meantime, attention to words becomes more selective: for example, in Head 8 of Layer 10 CLS attends wither to some adverbs (”perhaps”, “maybe”), or pronouns, or verbs, but the exact focus depends on the particular text. When attending to names in higher layers, CLS more often focuses on a particular name instead of attending to all names in the pair equally.

Such selectivity may indicate that a certain type of feature selection is happening in the later layers. Features on different levels, including subword, word, and syntactic ones, could be extracted and represented in the embeddings throughout the lower labels, while in the higher ones CLS attends more to specific features that are most relevant in this segment. Such variability is explainable given that different segments display very dissimilar components of storytelling: features that are most useful for dialogues may not be as relevant for expository or depictive narratives.

Interestingly, this type of behavior resembles the traditional authorship attribution pipeline (Marusenko, 1990), in which feature extraction is followed by the selection of the most distinctive features for a particular set of investigated texts.

8.3 CLS attention as means of retrieving tokens relevant for classification

After exploring all attention matrices, we selected the combinations of layers and heads in which relations seemed to be the most relevant and analyzed their distribution and key properties.

8.3.1 Distribution of relevant attention with respect to CLS

We observed that the attention to CLS from other tokens and the attention from CLS are unevenly distributed across layers, as shown in Figure 8.11. More specifically, tokens strongly attend to CLS in the lower layers 1–3, with this attention being variable, selective, token-specific, and meaningful in that it creates stronger links to CLS for tokens that were shown to be important for the prediction.

After layer 3, attention to CLS substantially decreases and becomes weak and mostly uniform for different tokens. In the meantime, CLS starts attending tokens other than itself and SEP, and towards the middle layers, this attention grows stronger, denser, and wider in terms of the scope and the diversity of attended tokens.

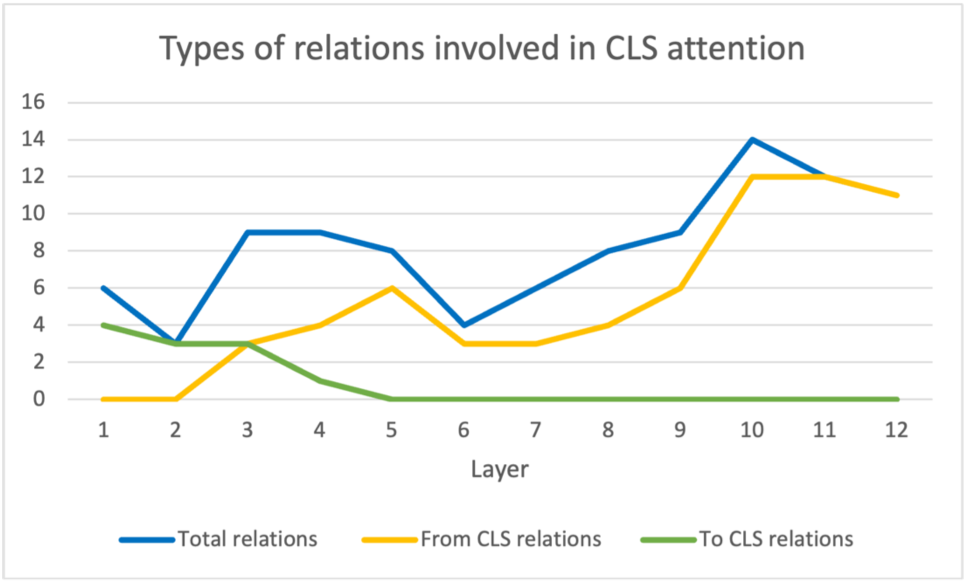


Figure 8.11: *Number of times a particular category of attention is observed at each level*

In the higher layers, another tendency for the attention outcoming from CLS arises: while getting more sparse, it begins to show text-specific character: in the same head, CLS may attend to different types of tokens in different texts.

We also observed some general trends reported by (Jawahar et al., 2019) re-emerging in the fine-tuned BERT. In early layers, many tokens indeed attend to CLS; in middle layers, the attention directed at SEP grows, and in the higher layers a lot of attention is directed in punctuation. This may indicate that, in general, fine-tuned BERT follows a similar procedure for encoding different aspects of the text.

8.3.2 Selective coverage in CLS

Another important property of attention from CLS is its selectivity in terms of the region of the text, shown in Figure 8.12. Attention from CLS is typically not limited to a particular context window: it can reach the tokens that are selected at a particular layer and head within a global range, regardless of their distance from the CLS token, which always comes at number 0. However, the analysis of attention maps showed that in many heads attention from CLS is actually directed at tokens from a particular text from the pair. This limitation can be strict (when no attention is directed to the tokens beyond the separator) or loose (when certain attention gets outside a single text, but it is weaker or more sparse).

Such behavior is understandable, given that the goal of the CLS token is to accumulate information relevant to the classification task for which the model is fine-tuned. For authorship verification, the ultimate goal is to represent stylistic differences between the texts to be able to conclude whether they are written by different authors. Therefore, at certain stages, CLS may be aggregating various information about each text separately in order to be able to compare them in the later layers, which are claimed to be the most task-specific (Rogers et al., 2020) and build the final representation.

Indeed, that proportion of equal and one-text attention is observed throughout the layers. The attention that is specific to a particular region starts at layer 4 (earlier CLS attention treats both texts equally) and peaks at layer 5, coinciding with a decline in the number of attention types that consider both texts. This drop of equal attention types happens in the middle layers, which are considered responsible for short-distance syntactic relations. Towards the higher layers, equal types of attention become dominant again. Interestingly, attention directed at the second sequence is typically more popular than that directed at the first one.

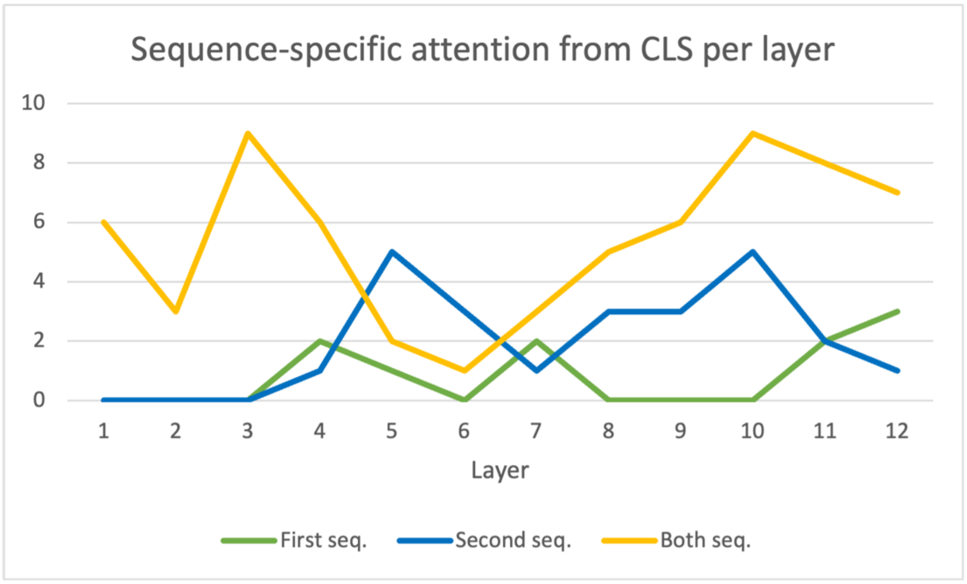


Figure 8.12: *Number of times CLS attends to tokens in first, second, or both sequences at each layer*

8.3.3 Types of tokens attended by CLS

For each layer and head, we generalized the current type of attention relevant to CLS and described it in terms of the types and categories of tokens being involved.

For the categories, we used the most general description that indicated whether the attended token is a content word, a function word, a punctuation mark, or a subword token. Importantly, we differentiated subword tokens as a separate category only in cases when they represented a subword-level feature, such as a grammatical affix. When attention was directed at all names, including those segmented into subword tokens, we considered it a single type of attention that is directed at names, that is, at content words.

Figure 8.13 displays the proportion of main types of tokens involved in CLS attention (not only attention from CLS, but also to CLS, and that between tokens relevant to CLS) in each layer. To visualize that proportion, we counted how many times a relation involving that type of token was observed in all heads of a given layer. For example, in layer 1, which had 6 CLS-related types of attention in total (see Figure 8.12), content words are involved in 6, function words – in 1, and subword tokens – in 4 relations.

Note that the total counts of categories do not have to add up to the total number of relations, as one relation may involve multiple categories. This is particularly common for attention from CLS, which may be directed at a large variety of tokens, while still being highly selective.

It can be seen that subword tokens are predominantly involved in relations in the earlier layers. Content words remain relevant throughout the whole model, usually appearing in more than half of the relations. Their share slightly declines in layers 3–5, when punctuation and function words arise.

Punctuation is not involved in CLS-related attention until layer 3, which goes in line with observations of Lin et al., 2019 indicating that the lower layers mainly collect information about the linear word order and typically do not involve syntactic relations, for which attention to punctuation may be required. Involvement of punctuation in CLS-related attention grows at layer 4 and further increases in the later layers, in which CLS continuously attends to full stops. The drop in attention of CLS to punctuation in the middle layers, which are considered responsible for most syntactic relations, can be explained by the fact that these relations first need to be extracted from the text and accumulated. This is presumably happening in layers 6–8, in which we discovered multiple types of attention between punctuation, or between function words to punctuation, or with punctuation being attended by words within one sentence.

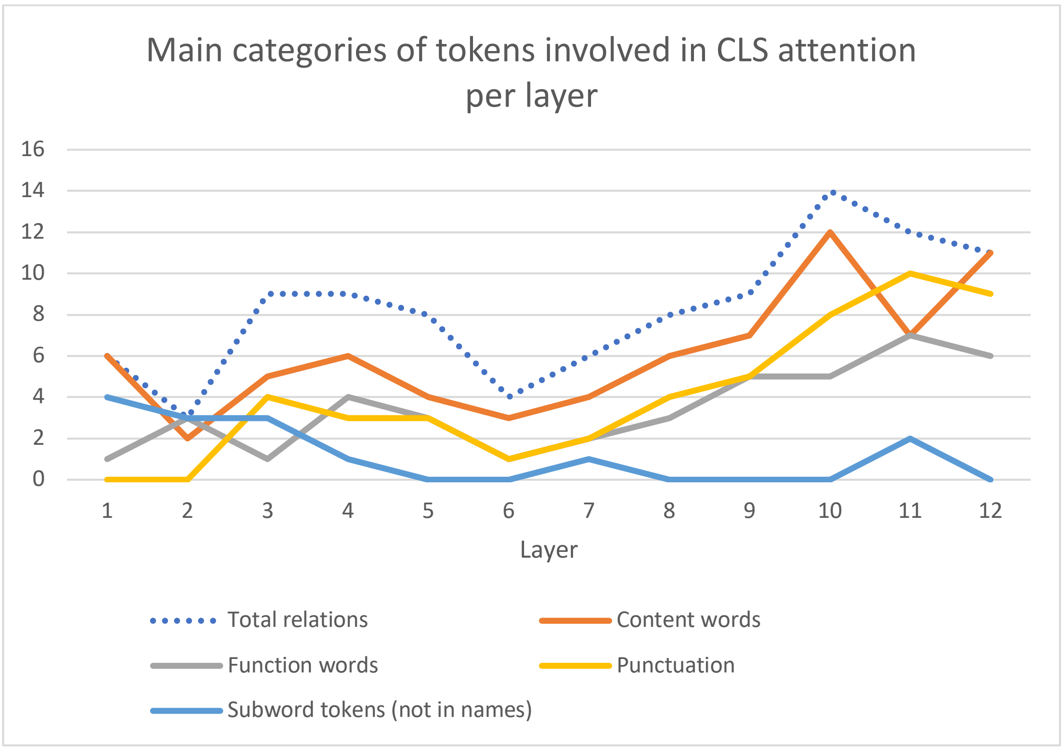


Figure 8.13: *Number of times each of the main categories of tokens is involved in CLS attention at each layer*

8.4 Highly attended tokens as linguistic features

We proceeded with analyzing the general statistics of the retrieved highly attended tokens. For each pair of texts out of 100 samples, we analyzed 64 selected attention maps that were shown to link CLS with tokens relevant for style formalization. We then extracted from these attention matrices the tokens with attention exceeding the threshold (0.01), sorted them by attention strength, and selected the 40 most highly attended tokens for each text pair from our sample. Such filtering of only the most highly attended goes in line with the suggestions of Jain and Wallace (2019) that only top-k attention weights are important and enables us to filter out the manifold tokens that are attended by CLS with only a residual weight, yet can end up counted as frequently attended due to their overall frequency in language.

The threshold value was chosen as a trade-off between the manifoldness and representability of the resulting features set. We observed that with values over 0.02 only a few most common tokens, similar in most texts, were selected (mostly commas and full stops), while with values smaller than 0.005 the filtering barely reduced the number of features. Thus, we have chosen the value within this stretch as a starting point for important features extraction.

As a result, we obtained 818 different tokens out of 4000 total counts.

Punctuation

1. Overwhelmingly most frequent ones are **full stops**, which take almost 40% (1666 occurrences).
2. Other types of punctuation also appear among the most frequently attended tokens. These are, most notably, **quotes** and **commas** with 360 and 66 occurrences respectively.

3. Other commonly attended punctuation includes question and exclamation marks (32), semicolons and colons (18 and 15), closing parentheses (13), underscores (8), and hyphens (6).
4. **Special characters**, such as asterisks ("*"), tildes ("~"), and dollar signs ("$") are also highly attended, though their overall low frequency does not let them high on the list.

Verbs for reporting discourse

1. A highly notable group of attended words is that of **reporting verbs** (Carter, 2016) Among the conventional ones, the most frequent are: "said" (131 occurrences), "asked" (38), and "replied" (32). Less widely used but still frequently attended are "stated" (12), "yelled" (10), "countered" (9), "responded" (9), "explained", "commented", "questioned", "smirked", "answered", "agreed", "exclaimed", "whispered", "ordered", and more.
2. Some other frequent verbs, mostly **mental-state verbs** or verbs of motion related to speaking and communication, are not directly linked to reporting a discourse but are often used to describe the act of speaking and refer to an utterance or an act of inner dialogue conveyed using direct speech: "realized", "recalled", "thought", "nodded", "sighed", "lowered" (as in "lowered her voice"), "began", "continued", and more. These verbs constitute the majority of those highly attended.

Other verbs

Other influential verbs include those of motion ("moved" (11), "walked" (10), "sat" (9)), of sensory perception ("felt" (9)). In general, verbs that are not involved in describing a discourse are less frequently attended.

Function words

Another large group of tokens includes function words, namely:

1. **Conjunctions** ("and", "as", "that"), adverbs ("also", "then", "already", "immediately"),
2. **Prepositions** ("towards", "with", "by", "like", "to")
3. **Modal verbs** in full and short, negative and neutral forms ("could", "couldn")
4. **Auxiliary verbs** in various full and short forms ("was", "had", "d", "wasn", "didn", "m")
5. **Pronouns** are comparatively infrequent in this list. Personal first- and third-person singular pronouns in different forms are the ones that most commonly occur among highly attended. It can be explained by the fact that they are used to discriminate between first-person and third-person narratives and thus represent the stylistic preferences of an author regarding the perspective of storytelling.

Proper names and co-referring words

1. As was already observed, most names and proper nouns are also continuously attended, including both conventional single-token ones and those processed into multiple subword tokens.
2. Besides, we also observed a large variety of words used to refer to an individual or a group of people. They include
   a. words denoting people ("man", "person", "male", "female", "boy")
   b. means of addressing people ("gentlemen", "miss", "honey", "master")
   c. words denoting different relations ("friends", "siblings", "father", "partner")
   d. military ranks and titles ("admiral", "officer")
   e. ethnicity ("Jamaican", "Latino", "Jew", "Roman")

3. Various proper nouns that are not names of people are also attended. These are mostly real or fictional toponyms, and sometimes also names of organizations or institutions and works of art.

Abbreviations

Another group of highly attended tokens includes abbreviations (“HP”, “TV”, “CPUs”). Their inclusion by the model seems reasonable given that they might be indicative of the author’s background (authors more deeply familiar with a particular fandom, especially in those related to science fiction, may rely more heavily on technical abbreviations). Many commonly used abbreviations and acronyms also display substantial stylistic variation, and trends may change over time or in different dialects and sociolects. Besides, some abbreviations are specific to a particular fandom and are therefore more closely related to the category that includes names and topical nouns.

Numbers

Different types of numbers also commonly arise as highly attended. This includes both spelled numbers and digits, Arabic and Roman. Typical examples of influential numbers are years (“1990”), centuries (“XVII”), ordinal numerals used in names (“VI”), and some plain numbers as digits or words (“ten”).

Foreign words and dialect variations

Another interesting group includes:

1. Words in foreign languages (“Sie”, “Mädchen”, “adios”)
2. Loanwords with original spelling (“touché”, “naïve”).
3. A related group is the one that includes dialectal variations (“colour”, “favourite”, “limousine”, “endeavour”, “neighbourhood”, verbs ending with “-se” or “-ze”).

Infrequent words

Many infrequent words not involved in modeling the topic of the fandom are also attended. These are such words as “mystique”, “furrow”, “trundle”, and "mischievous”. Such words correspond to the category of vocabulary richness, which was often represented by measuring the number of hapax legomena (that is, words that only occur once) in the text. Given the limited size of each pair of segments, quantifying the number of rare words by means of hapax legomena is problematic. However, it can clearly be observed that in the model CLS often attends to words that are not part of a basic vocabulary and are therefore often segmented into subword tokens. Such subword tokens, even though they are not parts of names, still receive CLS attention, which is not the case for many subword tokens used for grammatical affixes.

Errors

Several types of errors were found to be highly attended by CLS during the analysis of attention maps. Such mistakes include:

1. In-word errors (missing or redundant characters or inversion of characters)
2. Between-word (missing whitespaces or hyphens, missing function words, such as a determiner, or missing word from a set expression)
3. Casing mistakes (accidental irregularly uppercased characters).

Errors of certain kinds are not easy to spot among the highly attended tokens list since erroneously written words are segmented into subword-level tokens, and high attention of CLS may be directed at any of them. Missing words in a set phrase also cannot be represented in terms of important tokens: it cannot be reconstructed that the importance of a neighboring token is caused not by its own properties but by the absence of the one that commonly follows it. Therefore, for many typographic errors or missing words we can only rely on the observations drawn from the attention maps analysis. However, compound spelling (missing whitespace or hyphen between separate words) is easily retrievable, and we observe high attention frequently directed at such constructions ("nondiscrimination", "Thankyou", "getability").

Capitalized words

Finally, CLS often attends to capitalized words ("SERIOUSLY", "HECK") regardless of the part of speech. Partial capitalization, most frequently caused by a typing mistake ("CAme") is also attended.

8.4.1 Selectivity and causal interpretation

Causal interpretation on the level of particular attention layers and heads is hindered by the number of attention maps relevant to CLS. From 144 possible attention maps, we extracted 64 relations showing attention emitted from CLS, and that attention was selective in all cases except one, in which an earlier layer (layer 3, head 5) showed relatively uniform attention from CLS. The scope of this selectivity is rather complex: CLS in different heads may attend to various combinations of features, with different emphasis on each component of such combination. For example, attention to content words, such as names, cannot be distinctly separated from attention to punctuation because in multiple heads CLS attends to both. Moreover, selectivity in names can also be different: some heads direct CLS attention at a particular text-specific name, while in others attention involves names, pronouns, and co-referring nouns or noun phrases. Finally, similar attention in different heads may be directed at different sites of the text pair: there are those that strictly limit the range of attention to one text and those that attend tokens in a particular text from the pair with higher intensity.

In all these relations, CLS typically attends to other tokens with fairly moderate strength, especially compared to the intensity of outcoming attention in other tokens. Therefore, each attention link from CLS to a certain token does not change its embedding substantially, given that limited weight.

Thus, separating a particular layer-head combination that enables the model to arrive at a particular is complicated. The process can rather be described as a gradual construction of the CLS embedding, which is relatively equally distributed between heads, with a slightly bigger contribution of heads 4 and 12. It starts in layer 3 and grows until layer 5. In these early layers, there are also numerous relations that involve tokens that are attended by CLS in current or subsequent layers. Seeing a small decline afterward in the middle layers, it then regains its importance closer to the final layers, in which CLS attention becomes more focused on full stops and also more selective in terms of the types of tokens it attends in different texts and the range of attention.

8.5 Comparison with other known features

8.5.1 Overlap with features extracted by Boenninghoff et al.

Boenninghoff et al. (2019) proposed an explainable model for attribution that used attention modules to link different layers of bidirectional LSTMs. The authors then extracted linguistic features from 100 samples taking high attention weights into account. This is a different case of utilizing attention weights as an explanation for predicting authorship. Firstly, this model performs feature extraction on two texts separately instead of considering the pair jointly. Secondly, the encoded input does not encode tokens used by BERT – that is, CLS and SEP, and analysis is based on relations between tokens instead of checking to which tokens a particular classification token typically attends. Nevertheless, we find the list of features extracted and exemplified in their work highly important and illustrative and therefore provide a discussion of differences and similarities between extracted features.

Among the "Punctuation" features, we did not specify "accumulation of punctuation marks", which primarily involved ellipsis (three dots). In terms of syntactic properties, our model treats three dots as a single full stop – that is, words from the sentence, or other punctuation, or other tokens involved in representations of syntactic relations, attend to one of the dots (typically the first one). However, we noted the involvement of ellipsis in two between-token types of attention, observed in layer 8 (head 2) and in layer 9 (head 3), as we discovered specific links that only occurred in a sequence of dots and not from a separate full stop.

Besides, we highlighted the particular importance of quotes and their frequent connections with reporting verbs and names or co-referring constructions, while in (Boenninghoff et al., 2019) quotes are not marked as important punctuation. However, this difference is related with the different type of texts used for attribution.

In the "characters" features, we did not differentiate between variants of typing errors: all erroneous are processed into subword tokens, and these subword tokens are typically highly attended regardless of the particular type (missing, switched, or redundant characters)

We also discovered the strong importance of casing and described fully capitalized words and erroneous capitalization of particular characters as separate features. However, among the extracted tokens there were not many examples of using lowercase when uppercase is conventional, and we did not define lowercase as a discrete category.

Erroneous compound spelling is also described in our work, though as part of a wider group of between-word mistakes, which also includes missing words in set phrases. Acronyms and abbreviations are taken into account in our analysis. However, we did not differentiate between various types since we noted that both commonly used and unconventional or topic-specific abbreviations attract high attention. Both diatopic variations and foreign words were commonly listed among the commonly attended tokens.

In terms of stylistic features, in which the authors group a rather diverse set of features, we observed the importance of colloquial and obscene words. However, we did not note alternative spelling (when a dialectal alternative existed, we attributed it to diatopic variations, and when the alternative spelling was unconventional, it was processed as a regular mistake). We also did not specify neologisms as a distinct category. Many unusual words that are unknown to the tokenizer were not introduced by that author but are typical for a particular fandom. Differentiating between such topical words and pure neologisms would be unfeasible without a deep understanding of each fandom in the sample. Finally, we did not generalize a group of unusual interjections and discourse particles. Interjections are indeed sometimes attended by CLS together with other function words or stylistically variable or optional words, such as adverbs or prepositions for which there are commonly known alternatives. However, we did not find a particular attention map in which interjections are selected solely or with particular emphasis. Therefore, we ascribed them to a general category of stylistically variable words.

We did not extract any specifically syntactic features that are noticeably selected by CLS. However, we noted specific attention maps that selected clauses, explanatory phrases, or other distinctive syntactic constructions. Besides, we observed intense and variable attention to function words, with particular intensity of attention to words for which different short and full forms exist ("did" and "didn", "have" and "ve").

The strong attention to proper nouns was widely highlighted in our analysis, with a more detailed observation of the potential influential properties of names. Besides, we observed the importance of other words used to refer to people or address them. However, we did not consider mistakes in the spelling of the names as a separate feature, given that the reconstruction of correct forms is problematic for unconventional fictional names.

Thus, out of the 29 features outlined by (Boenninghoff et al., 2019), we observed and described 11 due to a more coarse-grained categorization and a different character of analyzed texts.

Besides, we extracted numerous linguistic features not listed by the authors. These are, most notably, verbs related with narration, quotes, infrequent words, words with stylistic variation, short and full forms of function words, words used to refer to people and address them, and numbers. Naturally, many of them are specific to the dataset on which the model was fine-tuned: reporting verbs, quotes, and words referring to people are most common in fictional literary works with a certain plot and set of characters, and such features cannot be expected in a corpus of non-fiction literature or social media microtexts.

8.5.2 Overlap with features commonly used in authorship attribution

Features typically used in machine learning models trained for authorship attribution are typically derived from the raw input: these can be word embeddings or counts of words and n-grams on word- or character-level, possibly weighted by TF-IDF. Such features are not directly comparable to the ones that we extracted from the attention maps since those are represented as separate tokens.

However, several features show a notable resemblance to specific parameters attended by our model. Among them is, firstly, vocabulary richness, which is often formalized as the number of hapax legomena (words occurring only once) and dis-legomena (words appearing twice). This parameter was used as one of the "Style" features by Sari et al. (2018) and as one of the "Lexical" features in Stamatatos (2009). We suppose that the high level of attention of the model towards rare words of all parts of speech can be related to this metric, as their rarity implies the range of vocabulary that an author possesses.

Other "style" features used by Sari et al. that correspond to those in our model may include the counts of punctuation (since the attention of CLS to all punctuation in the sentence may involve counting), of function words, and of digits (since we observed that numbers are frequently attended, though not only in the form of digits, but also as roman numerals or words). They are shown in table 8.1.

Type	Group	Category	#	Description
Style	Lexical	Word-level	2	Average word length, number of short words
		Char-level	2	Percentage of digits, percentage of upper-case letters
		Letters	26	Letter frequency
		Digits	10	Digit frequency
		Vocabulary richness	2	Richness (hapax-legomena and dis-legomena)
	Syntactic	Function words	174	Frequency of function words
		Punctuation	12	Occurrence of punctuation
Content	Word *n*-gram	Words unigrams	100	Frequency of 100 most common word unigrams
		Words bigrams	100	Frequency of 100 most common word bigrams
		Word trigrams	100	Frequency of 100 most common word trigrams
Hybrid	Char *n*-gram	Char bigrams	100	Frequency of 100 most common character bigrams
		Char trigrams	100	Frequency of 100 most common character trigrams

Table 8.1: *Types of stylometric features (from Sari et al. (2017))*

Among the set features generalized by Stamatatos (2009), shown in Table 8.2, we note errors (though in that work they are classified on a lexical and syntactic level, while we differentiate three common types of observable errors, including character-level ones, and mention other types of errors in ablation experiments with LIME).

Part-of-speech data also seems to be involved, given that the majority of highly attended tokens fall into a limited number of categories. However, for most parts of speech, the model favors a certain type of words: verbs describing discourse, nouns referring to people, auxiliary verbs with variable forms, and adverbs with semantically similar analogs. Therefore, we argue that the model does not rely solely on part-of-speech counts.

Synonyms, categorized as a semantic-level feature, are also taken into account to a certain extent by high attention directed at words with substantial stylistic variation, which implies that they can be replaced by a close synonym without a meaningful change of semantics, and the choice between such synonyms is therefore subject to individual stylistic preferences.

Features		Required tools and resources
Lexical	Token-based (word length, sentence length, etc.)	Tokenizer, [Sentence splitter]
	Vocabulary richness	Tokenizer
	Word frequencies	Tokenizer, [Stemmer, Lemmatizer]
	Word *n*-grams	Tokenizer
	Errors	Tokenizer, Orthographic spell checker
Character	Character types (letters, digits, etc.)	Character dictionary
	Character *n*-grams (fixed length)	–
	Character *n*-grams (variable length)	Feature selector
	Compression methods	Text compression tool
Syntactic	Part-of-speech (POS)	Tokenizer, Sentence splitter, POS tagger
	Chunks	Tokenizer, Sentence splitter, [POS tagger], Text chunker
	Sentence and phrase structure	Tokenizer, Sentence splitter, POS tagger, Text chunker, Partial parser
	Rewrite rules frequencies	Tokenizer, Sentence splitter, POS tagger, Text chunker, Full parser
	Errors	Tokenizer, Sentence splitter, Syntactic spell checker
Semantic	Synonyms	Tokenizer, [POS tagger], Thesaurus
	Semantic dependencies	Tokenizer, Sentence splitter, POS tagger, Text Chunker, Partial parser, Semantic parser
	Functional	Tokenizer, Sentence splitter, POS tagger, Specialized dictionaries
Application-specific	Structural	HTML parser, Specialized parsers
	Content-specific	Tokenizer, [Stemmer, Lemmatizer], Specialized dictionaries
	Language-specific	Tokenizer, [Stemmer, Lemmatizer], Specialized dictionaries

Table 8.2: *Types of stylometric features (from Stamatatos (2009))*

8.6 Overlap with LIME

8.6.1 Feature comparison

For each of the 100 samples, we obtained the list of highly attended tokens. These lists contained multiple repetitions due to the fact that attention is calculated for each of the identical tokens (such as full stops) separately, and attention in different layer-head combinations may highlight the same token multiple times. We filtered out SEP and CLS tokens and reconstructed full words from the subword tokens, so that all subword units receiving attention from CLS could contribute to the weight of the word to which they belong.

Raw counts of all highly attended tokens, sorted by the attention weight emitted by CLS at an individual token, did provide meaningful insights into the variety of important features employed by the model when we considered general statistics for all text pairs. However, they obscured the importance of each feature for a particular pair: it was not clear whether a large number of occurrences of a token as highly attended implies its importance for the prediction in question, or all these occurrences only had a weight slightly above the threshold, and it is the overall frequency of this token in the dataset that ensured its many counts.

To unveil this difference, we exploited two ways of selecting the most important tokens for each text: we first considered the 20 tokens that are most frequently attended by CLS (combining the counts for identical tokens), and then – the 20 tokens that receive the highest attention weight, with the weight being accumulated for all identical tokens. We intentionally used a larger number of extracted features for the attention-based explanation compared to LIME (20 and 10 respectively) to account for punctuation, numbers, and special symbols, which are not contained among the LIME features and are commonly selected as important by the attention analysis.

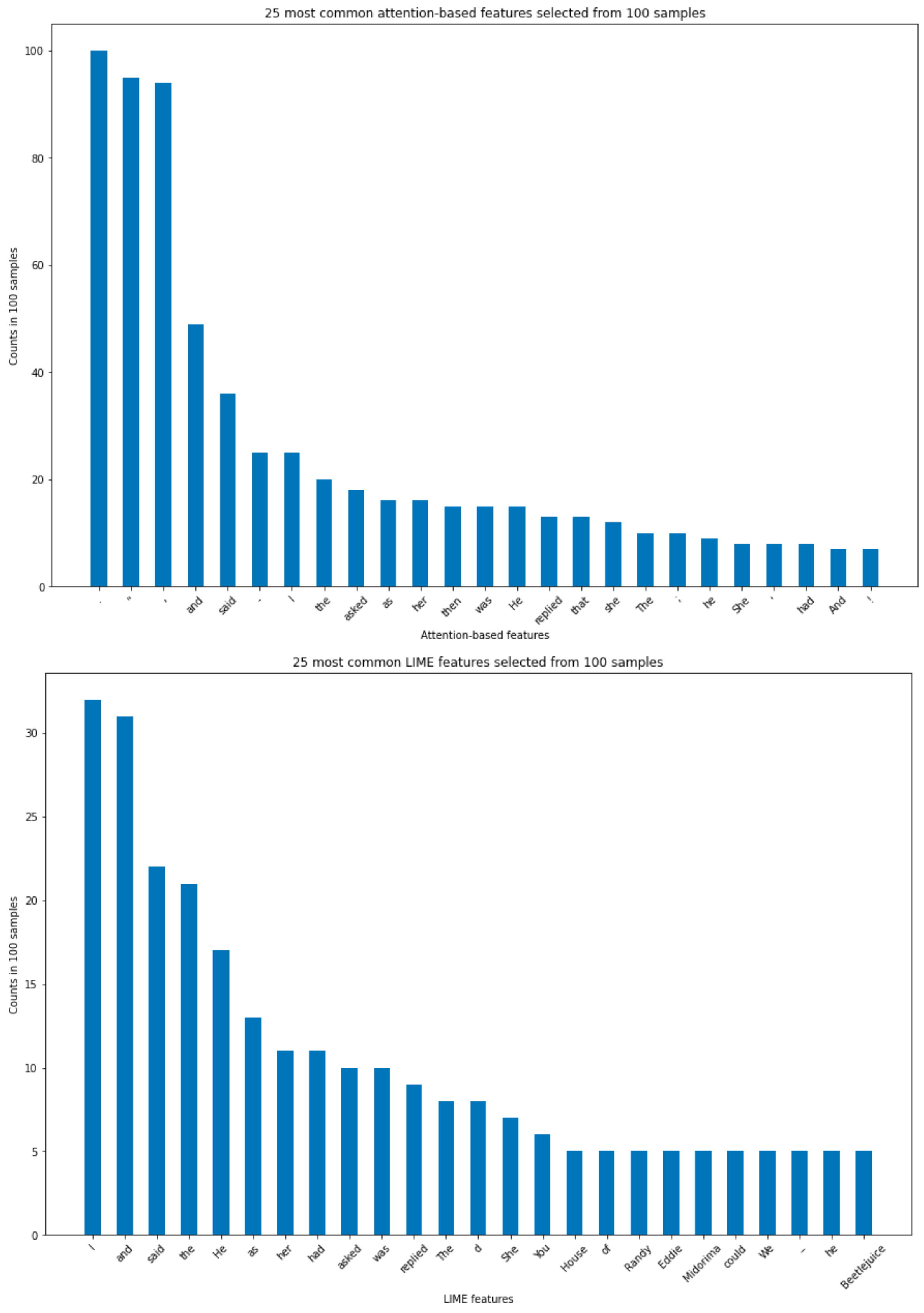


Figure 8.14: *Counts of 25 most common features selected in all 100 samples by attention-based procedure (top) and LIME (bottom)*

In Figure 8.14, the 25 features most commonly selected by an attention-based algorithm using counts and by the LIME explainer are visualized. It can be seen that the most salient difference is the overwhelming importance of punctuation in attention-based features, which is inevitably absent in LIME. At the same time, various other word-level features in different categories overlap: in both cases, we observe pronouns, reporting verbs, and conjunctions. To better understand the difference between the two approaches, deeper analysis of features beyond 25 most commonly selected was performed

	Counts, top 20	Sum, top 20
Overlap	0.554	0.548

Table 8.3: *Overlap between features extracted using attention-based procedure aggregated using counts or sum and the features extracted by LIME*

As shown in Table 8.3, we observed that over 55% of important features that are selected by LIME are also present in the list generated based on the counts of times CLS attended that type of token. The number for the list that is based on the total attention emitted at that type of token by CLS is comparable (54.8%). Moreover, in 20% of instances the overlap reached around 80%, which means that 8 out of 10 LIME features could also be found among the top-20 most heavily attended features.

The words that were selected by LIME but did not receive enough attention to be selected by the attention analysis algorithm vary for different text instances, but noticeable categories include discourse particles (“Ahhh”, “Ohhhh”, “Oh”, “Yeah”, “Damn”), multiple non-reporting verbs, pronouns that are not first- or third-person, determiners (typically “the”).

Besides, some words naturally appear on the list due to different tokenization being used: words that are erroneously written without whitespace, or those preceded or followed by an underscore, are processed by LIME as a single token, while the BERT tokenizer divides them into several tokens.

There are also accidental occurrences of words from categories that we selected as important: dialect variations, numbers, first- and third-person pronouns. This may be explained by the fact that we had to limit the number of important tokens extracted based on CLS attention by 20, which resulted in some important tokens being missed. This unveils the explainability trade-off between the simplicity and completeness of explanations.

8.6.2 Feature importance

To understand whether the model actually treats such infrequent but heavily attended tokens as important and relevant for the final prediction, we compared the mean importance of selected tokens for each of the selection techniques. We calculated the importance of each token I_t as the mean of absolute values of the difference between the original logits L and logits predicted after removing that word from the text L' for each class c out of N — that is, for classes 0 and 1.

$$I_t = \frac{\sum_{c=0}^{N} |L_c - L'_c|}{N}$$

Revealing how large the importance value is compared to the original logits required calculation of the average magnitude of the mean of logits for two classes in each instance, which was 4.71. The results are provided in Table 8.4.

For the counts-based selection, which showed the highest resemblance to LIME feature extraction, the mean importance of 20 selected tokens was 0.804. Given the mean magnitude of logits, these features on average account for over 17% of each logit (both positive and negative). Along with the average importance of top-20 tokens, we also selected one token with the highest influence on the prediction logits for each instance and found the average weight of the most important token in each instance. That value was 4.49, constituting over 95% of the mean logits.

In practice, it means that removing that token from the text on average results in total confusion of the model, as both logits are pushed towards 0. For the values even slightly above average, removing such a token will lead to the inversion of the prediction, as the logit with the highest value will change.

These #1 tokens for each text are mostly full stops (19/100), commas (10/100), quotes (8/100), first- and third-person pronouns ("he": 3/100 and "I": 2/100), conjunctions ("and", "as": 2/100 each), verb form "said", various names and co-referring nouns, special characters, and words with stylistic variants. It appears that more complex names, including foreign and compound ones ("Sumireko", "Zexion"), are more often highlighted as the most important tokens in the text.

For comparison, we also calculated feature importance in a similar way for features extracted LIME. It can be seen that the mean importance of all features selected for each text is higher, contributing on average 0.982, or 20.85%, to the average value of logits. This can be explained by the fact that fewer features are selected by LIME (10) compared to attention analysis (20), and given that LIME arranges features by their estimated importance, such difference can be expected.

Nonetheless, the mean importance of #1 most important token is significantly lower, constituting 3.626, or 77% of an average logit. This means that, when LIME features are used, the most important token on average does not decrease the model's certainty to 0, as almost a quarter of an original logits value for each class is preserved.

These results suggest that, even though LIME extracts fewer features that are slightly more influential upon averaging, it is less capable of detecting the heaviest instances of the text in terms of prediction change. This motivates the use of an attention analysis algorithm for probing the robustness of similar models for authorship attribution. For the list of features, a combination of LIME and attention-based techniques can be used for wider coverage.

The comparison of two extraction techniques shows that, even though high attention is accumulated in certain tokens, relying on them instead of more frequent ones does not benefit the explanation capacity of the algorithm, and the most efficient technique is also the most straightforward one, in which tokens are selected plainly based on the number of times it was attended by CLS with intensity above the threshold.

	Mean value	Mean value as % of mean logit
Mean importance of top-20 tokens, Counts	0.804	17.07%
Mean importance of top-20 tokens, Sum	0.782	16.6%
Mean importance of top-10 LIME words	**0.982**	**20.85%**
Importance of #1 token, Counts	**4.490**	**95.33%**
Importance of #1 token, Sum	4.346	92.27%
Importance of #1 token in LIME	3.626	76.99%
Mean logits magnitude	4.710	100%

Table 8.3: *Importance of top-20 and top-1 tokens extracted by LIME and attention-based procedure, compared with average logit magnitude*

8.6.3 Label inversion

The high explanation capacity of the features selected based on counts goes in line with the observations of the number of label switches upon feature ablation. For each important feature, we considered not only the difference between the original logits and logits predicted after removing that feature but also the final label and registered the cases when the label was not equal to the originally predicted.

As shown in Table 8.4, on average, in 6.8% of extracted important features (that is, in 136 features) removing solely that feature led to an instant label switch. For all text pairs, at least one feature out of 20 selected could achieve a label switch in 50% of cases, which seems to be reasonable given that a label switch requires a change of logits greater than their original value and average logits magnitude is very close to the average weight of the most important feature in the list.

Switches, token-wise	6.8%
Switches, instance-wise	50%

Table 8.4: *Percentage of label switches for the attention-based procedure with count aggregation*

8.7 Names as features

8.7.1 Overview

Names repeatedly occur both among the most commonly and the most intensively attended tokens. They are also frequently selected as important features by LIME, and input obfuscation confirms their importance, as removing or altering names often results in a significant change of logits or even a switch of the predicted label.

The analysis of attention maps in various layer-head combinations shows that this is not a plain result of a linear combination of attention directed at subword tokens contained in names. In many heads, single-token names are attended concurrently with composite ones, and selectivity of the attention emitted from CLS most commonly leads to only one or two subword tokens from each word being attended. Besides, single-token names are often shown to cause a significant change in the prediction upon altering, and no direct correlation between single- or multi-token names and their importance was observed.

To retrieve a broader view of the influence of names on the prediction, we performed a different type of obfuscation for those highly attended tokens that were identified as names.

Preliminary experiments showed that their behavior may differ. Some names show the behavior close to that of linear features: increasing and decreasing the number of their occurrences correspondingly changes the prediction. Others turn out to be sensitive to the casing to an even larger extent than to changes in the name itself. In other words, a name can act as a different type of feature, with different levels of analysis involved. Defining the relevance of potential types required a separate experiment.

8.7.2 Names ablation and alteration techniques

Based on the previous observations, we outlined a number of possible ways in which names can be involved in the classification of a pair of texts.

Plain ablation

Firstly, a name in a particular context may plainly serve as a keyword for memorizing a particular segment. Such shortcuts can be used by the model in particular author classes, which are significantly biased towards one label, or only incorporate texts from a certain fandom, in which there is a high probability of that name appearing. In this case, we hypothesize that removing that word from the surrounding context should result in a significant change of the prediction logits, likely also causing the inversion of the final prediction. At the same time, adding more identical names should not influence the prediction substantially, as it does not affect the name-as-keyword occurrence.
To retrieve such cases, we performed deletion of names from the text, as in the previous experiment with all types of tokens.

Multiplication

Secondly, a name can act as a linear feature. Should this be the case, doubling the number of occurrences will create the difference between original logits and logits after multiplication proportionally to the original importance of that name, which can be established by deletion.

Such behavior would mean that the name is not that important on its own, and it is rather its frequency in the text that defines the author's style.

Replacement

However, it is yet unclear in such setting whether it is this particular name that is important for the model, or any name standing in a certain position and surrounded by a particular context.

Therefore, we perform multiple alterations to address that issue.

We replace each name with a conventional male and female name in conventional spelling ("John" and "Mary") to see whether a new name leads to a substantially different behavior of the model and whether any patterns can be observed separately for a male and a female name.

We also replace the name with an unconventional and fandom-specific name ("Rinoa") which is tokenized into a list of subword tokens to see if the impact of such alteration is larger.

Uppercasing

Moreover, aside from the quantitative features, names may also be related to features representing typographic variation. Therefore, we assess the importance of casing by replacing each name with its uppercased and lowercased versions. If it is the way each name is spelled that is important for the prediction, such changes can be expected to result in significant prediction alteration.

Advanced replacement

Finally, the name can be important as long as it fulfills its role as a reference to a named person. If this is the case, replacing names with pronouns or other co-referring words should not influence the prediction substantially, as it preserves the reference to that character. To assess whether this type of behavior takes place, we replaced names with pronouns ("she" and "he") and a gender-neutral noun phrase referring to the character ("the person").

This type of feature stretches beyond a purely linguistic analysis of the text, as it indicates the structure of the narrative by telling how often a particular character is being involved, regardless of the particular verbal expression used for such reference.

There are, of course, other features representing the structure of the narrative that are more or less directly related to names. After all, names are used to refer to the characters involved in the story. Not only the order and intensity with which they are involved but also the exact set of characters selected for a particular act of storytelling may well be characteristic of a particular author. However, such properties lie beyond a purely stylistic analysis of the text and, therefore, beyond the scope of the current research. Besides, analyzing them requires thorough labeling, which is further complicated by the rather exotic topic.

Even labeling the character's gender, when the names used in the text are unconventional and the pronouns are not always available in the same segment, is non-trivial. Additional insights can be drawn from a more detailed description of each character in terms of a particular literary work and fandom. Such description may include an indication of whether that person is conventionally a protagonist or antagonist, a main or a minor character, whether it is a well-known figure, or one from the extended universe, unknown by the general audience, or completely made up by a particular author.

8.7.3 Names ablation and alteration results

After retrieving 513 names from 100 segment pairs, we performed 11 input obfuscations described above. We then obtained the absolute values of changes in logits after feeding the altered text into the model and took the mean of these changes for both classes. The resulting value was used as means of representing the importance of each alteration. For these values, we calculated the mean and standard deviation, reported in Table 8.5.

Type of change	Mean influence on logits	Standard deviation
Del	1.012	1.855
Upper	1.133	2.060
Lower	1.038	2.011
Duplicate	0.355	0.711
John	1.059	1.943
Mary	1.072	1.964
Rinoa	**1.323**	**2.157**
He	0.909	1.750
She	0.943	1.766
The person	0.968	1.784
The boy	0.976	1.837

Table 8.5: *Mean influence of changes in names on logits*

In general, all types of changes in names except one show relatively high influence on prediction logits compared to the average influence of top-20 features selected based on counts. However, it can be seen that multiplying the number of occurrences of a name in question is the least important change: on average, it only changes the logits by 0.35.

The boxplot visualization in Figure 8.16 also shows that the median values differ from the mean by a large margin, which can be explained by a number of outlying features which show very high importance with respect to the prediction logits. For most types of alterations, some tokens showed changes in logits by over 9 (which is over 200% of the average logit magnitude). Only for multiplication, most of the outliers did not diverge from the median value by over 2.

The substantial number of outliers together with the low median value indicates that not all names are equally important for the model: there are some with extremely high influence and the majority which does not substantially differ from other features (for which the average influence is 0.8).

Deletion, which had been used for all types of tokens in the previous experiment, shows an average logit change close to 1.0, and most other features perform comparatively close to that value, though some differences can be noticed. We pay particular attention to changes that result in a smaller average change in the prediction logits than a regular deletion of a name, as it indicates that certain information that used to be encoded in that name can still be encoded after replacing it with an alternative.

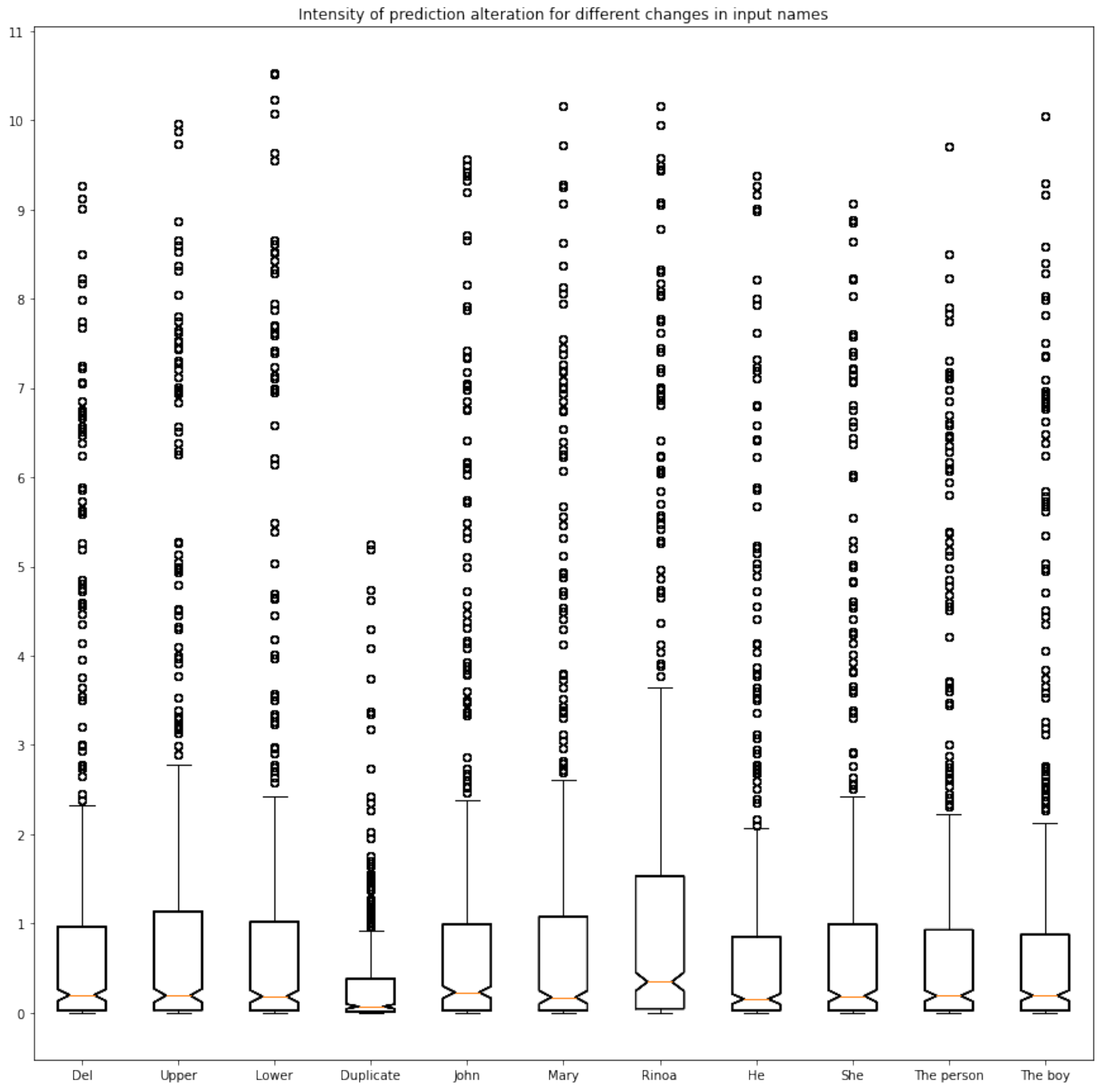


Figure 8.15: *Influence of changes in names on logits*

Converting the whole name to upper case is consistently more important than converting it to lower case, with uppercasing being the second most important change overall. This can be explained by the fact that the upper case can be used as a device for conveying expressivity. For example, it can be an indication that the name is shouted or otherwise expressively. The lower case, on the contrary, does not provide such connotation by default. Instead, a name spelled without capitalization of the first letter is likely perceived as an error. A noticeable difference in their influence on the final prediction indicates that names are not only important on their own but also as means of conveying additional information about expressivity and emotionality.

In both conventional names and pronouns, replacing the name with a gender-specific alternative is consistently slightly more important when we choose a female name or pronoun as a replacement. Given the exotic character of names in many fandoms and the large number of non-binary characters involved in narratives, accurate labeling of the gender corresponding to each name is complicated, and we can only assume that this pattern is caused by the slightly unequal distribution of gender in characters in the texts, most likely with a higher percentage of male characters. When gender-specific pronouns and names are used to replace the original names, the gender of the character in question is inevitably switched for one of the two options, which causes substantial changes in the semantics of the text and results in a more significant difference in the final prediction. Therefore, the slightly bigger importance of female names and pronouns as replacements can be explained by the fact that, given a slightly larger amount of male characters in the data, such switches happen more often.

The difference after replacing the names with a gender-neutral construction (“the person”) and a gender-specific construction (“the boy”) turns out to be quite similar to that of the regular deletion, with a gender-specific noun phrase referring to a male child being slightly less influential, likely for the same reason as with the pronouns.

The pronouns themselves, when compared with conventional names, are less important by a comparatively large margin, indicating that names are not involved solely as means of referring to a person participating in the story, but are also meaningful features on their own. Besides, replacing names with pronouns undermines their relevance when a name is used by a person to address someone, to call someone by name (“Mary, look!”) rather than to refer to someone. This suggests that the type and quantity of such addresses may to a certain extent be involved in formalizing the style.

Finally, we also replaced each name with an unconventional name, which is typical for a particular fandom and is not part of the tokenizer’s regular vocabulary. We observed that the intensity of alterations in prediction was the highest for this type of change. This suggests that the inclusion of certain names is even more important than the name referring to a person with a particular gender. Such importance indicates that the coherence of the narrative and its agreement with the conventional range of characters for a certain fandom is still important and breaking it by adding an irrelevant character substantially changes the model’s behavior.

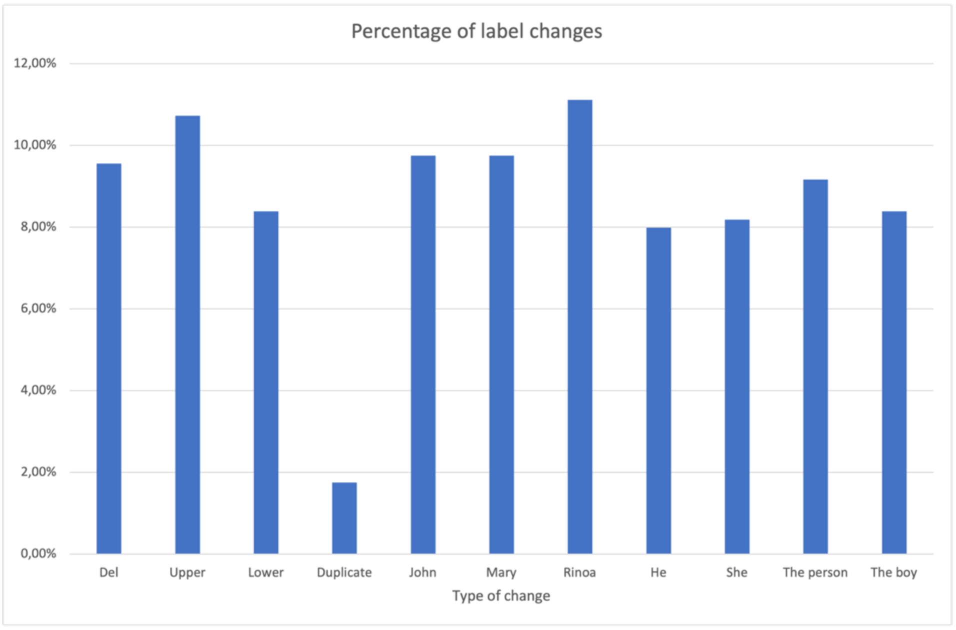


Figure 8.16: *Percentage of label inversion upon each type of change in names*

The mean changes in the prediction correspond quite well with the percentage of cases in which replacing a name with a certain alternative resulted in an inversion of the predicted label, shown in Figure 8.16. The number is the highest for an unconventional name (11,1%), closely followed by uppercasing (10,7%). Other conventional names and "the person" are roughly equally influential, switching labels in approximately 9.5% of cases, similar to the regular deletion. The pronouns and lowercasing are less important than deletion, resulting in label inversion in around 8% of cases. Lastly, duplication causes such inversion in a mere 1.75% of cases, which is an even more significant drop than that in average logits alteration.

9. Conclusion

In this project, our goal was to explore and propose a solution for authorship attribution based on deep learning architecture, namely a transformer model. That choice was motivated by their state-of-the-art performance in diverse domains of NLP and the ease of adapting such models to different datasets by means of fine-tuning without manual or semi-automated feature selection. The downside of such versatility is the problem of model explainability: the learned features and their influence on classification decisions are essentially part of a black-box algorithm. Although such opaqueness can be tolerated in some domains, authorship attribution can be used in forensic analysis or otherwise be related to reputational and financial risks, to which independent content creators are particularly sensitive. Therefore, for the current task of verifying the authorship of fanfiction texts, we considered explainability a desideratum. We focused on interpreting the model not only with the view to practical applications: understanding how the model functions and, in particular, on which features it relies, can help us better understand the value of existing stylometric parameters and the way complex stylistic representations arise in deep learning models.

Thus, our objective was twofold: it included solving the problem of authorship attribution by utilizing a pre-trained transformer-based model to verify that such models are able to learn stylistic differences and applying explanation techniques to uncover the process of making the classification decision and the features being involved. From that goal two research questions arise:

Q1 aims at observing whether *it is reasonable to utilize pre-trained Transformer models for the task of authorship attribution or their performance does not surpass that of smaller models* **Q1.1** formulates a specific research problem of Transformers for text classification: *What does the ability to process longer sequences contribute to the Transformer performance in context of AA?*

Q2 is formulated differently depending on our answer to *Q1.* If the answer is True (that is, a Transformer model for this task can be trained with sufficient prediction accuracy), **Q2-1** takes place: *Which meaningful parameters, if any, can we extract, and by which means?* Additional subquestion **Q2-1.1** concerns the nature of such parameters: *Do any of these patterns correspond to stylometric features used in the traditional stylistic analysis for attribution?*

This question corresponds with six possible hypotheses:

Hypothesis I implies that explanations related to existing stylometric features may be found by visualizing the attention weights of different heads at different levels

Hypothesis II aims at finding valuable features by generating explanations on fully connected layers at different levels

Hypothesis III suggests that the uppermost layer, the classifier itself, can provide sufficient explanations

Hypothesis IV involves explanations based on complex features (e.g. from combining multiple attention heads) observed, among other methods, using input permutation

According to *Hypothesis V*, explanations can be generated by combining information from the sources mentioned above

Hypothesis VI accounts for the negative answer, according to which no meaningful features that reveal causal relations between input and output and correspond to any of the existing stylometric features could be found

If the answer is False (and the Transformer model could be trained in principle but did not outperform the baselines), **Q2-2** is used to determine the possible causes of this issue: *Is this insufficient performance specific to our proposed solution, or related with the known bottlenecks of current Transformer models, or caused by some fundamental limitations of the Transformer architecture?*

In this case four hypotheses are in place:

Hypothesis I suggests that poor performance is due to limited input length of existing Transformer models that does not allow them to process the text as a whole and learn long-distance stylistic patterns, and therefore baseline models with limited input window size would also show a corresponding decrease in performance

Hypothesis II implies that the size of the pre-trained Transformer language model, i.e. the number of parameters or hyperparameters, constitutes a limitation, and a larger model could perform better in this task

Hypothesis III states that the general Transformer architecture is the reason for insufficient performance as it is incapable of properly learning quantitative features that are important for AA, and a better performance can be expected from a model that makes use of different architecture for the language model, such as LSTM-based one

Hypothesis IV covers the negative scenario in which none of these expected justifications can be proven.

9.1 Discussion of research question Q1

First, we planned to implement a model that can solve the problem of authorship attribution utilizing a pre-trained transformer-based model. To that end, we adopted the architecture suggested by Peng et al. (2021) and implemented it using commonly used and widely supported frameworks and tools. The model was evaluated compared to the common baseline and the existing BERT-based implementation to justify the use of such architecture. We then scrutinized various parameters, such as the limits of segmentation and the BERT input size, and trained two models: a base model with 256-token input and a large one that accepts 512 tokens.

We compared the performance of these models in terms of both predictions for individual segments and averaged text-level predictions and concluded that, even though the difference for separate segments is noticeable, it diminishes when these segments are averaged for the final decision. Considering that the large model takes significantly longer for all operations, including fine-tuning, predictions, and explanations, resolving the trade-off in favor of a slightly better performance does not seem justifiable at least for the task in question, in which comparatively long texts are available. Even more importantly, the analysis of attention maps of such length is a lot less feasible, and their visualizations often cannot be processed using the existing tools for displaying HTML.

These considerations enable us to **positively** answer **research question Q1,** which aims at observing whether it is reasonable to utilize pre-trained Transformer models for the task of authorship attribution, or their performance does not surpass that of smaller models.

We also collected sufficient evidence to answer the corresponding sub-question **Q1.1** which formulates a specific research problem of Transformers for text classification: *What does the ability to process longer sequences contribute to the Transformer performance in the context of AA?*

In this regard, we claim that the use of longer input is beneficial in principle in terms of evaluation metrics, but it hinders the usability of the model due to higher requirements. For longer texts, splitting into even more segments can be applied with a merely linear increase of training time and, as mentioned in Chapter 3, such technique provides additional benefits of assessing the confidence of the model by checking the percentage of correct segment-level prediction and of analyzing the co-authorship throughout the entire text by keeping track of the distribution of segment labels. For shorter texts, such as social media microtexts. increasing the input size, on the contrary, seems to be the most appropriate way of improving the classification performance since segmentation is unavailable or limited to a very small number of segments.

In a more global sense, the advantage of deep learning-based models for authorship attribution compared to alternatives is the ease of adapting it to a particular task: neither manual feature selection nor preprocessing steps, such as data annotation with a part-of-speech tagger, are required for fine-tuning.

9.2 Discussion of research question Q2

Secondly, we aimed at finding the meaningful parameters that can provide insights into the way the predictions are made in order to answer **Q2-1** (*Which meaningful parameters, if any, can we extract, and by which means? Do any of these patterns correspond to stylometric features used in traditional stylistic analysis for attribution?*)

To that end, we first provide a detailed investigation of the role of the final classifier in order to be able to disregard its explanations at the later stages. We then scrutinize local post-hoc explanations and analyze their relationship with the input characteristics and perform experiments with adversarial examples to highlight interconnections between the extracted features and other parameters.

After that, we proceed with analyzing the attention relevant to CLS and extract the tokens most highly attended at head-layer combinations marked as relevant as potentially important features. We classify these tokens into meaningful categories and describe them in terms of linguistic features to which they correspond.

Finally, we compare the most commonly extracted features in LIME and the attention-based approach to reveal the extent to which they overlap and to compare the importance of these features in terms of their influence on prediction logits. We conclude that, with over 55% of features overlapping, LIME and attention-based approach indeed extract largely similar features, with the exception of punctuation. While LIME features are more important on average, the most important features are more salient in attention-based approach.

Analyzing that procedure, we note that LIME provides an explanation algorithm that is interpretable with respect to the input. It is also model-agnostic: any model trained for that specific task of pair-wise attribution can receive explanations by a joint application of base LIME and LIME-pair. The problem is that it is non-deterministic: random permutations result in slightly different linear approximations. Such differences are not crucial for most salient features, such as some names, but lead to significant re-arrangement of less important ones. Therefore, it does not make sense to retrieve more than 10 features: they will be different in multiple runs. Even for 10 features, last 2-3 features are often dissimilar.

Besides, the existing version of LIME did not enable direct use of the BERT tokenizer because that tokenizer yields tokens encoded as integers, while the LIME text module necessitates the use of natural language strings. A character-level tokenizer, in turn, would not enable straightforward detection of the separator, which is needed for LIME-pair. As a result, extracted features did not include punctuation, which is a serious limitation and one of the most common reasons for the dissimilarity between the list of important tokens extracted using LIME and the attention analysis.

Attention analysis is driven by the model. It is partially transparent in that it uses particular attention matrices, which are known beforehand, and selects tokens most heavily attended by CLS from those matrices. However, the model's classification component is not taken into account. Due to numerous types of attention from CLS to other tokens, and a large variety of tokens being attended, we do not propagate weights of particular tokens attended by CLS through multiple layers and do not track their contribution to particular dimensions of the final embedding, and the influence of these dimensions on the classifier's decision. Instead, we rely on a general assumption that, given the overall sparsity, selectivity, and moderateness of CLS attention, all tokens attended by CLS in non-uniform types of attention contribute to a certain extent to the final representation. Given the goal of the explanation, which is not to provide an exhaustive interpretation of the model at all levels but rather to provide a motivation for the model's decision with respect to the input, we consider such an assumption reasonable.

This assumption is verified by the large percentage of tokens appearing both in LIME and attention-driven explanations (55%), which is drastically above a hypothetical chance level (the total amount of potential sets of important features, in the worst case, would be the number of combinations of 10 elements out of 256).

Another limitation of the attention-based explanation is that it does not directly link input with the output: instead of deriving the influence in the final prediction from some properties of the model and the input, we just select features that are likely to be important for the model based on the model's internal structure and then calculate the importance by input ablation and alteration.

In the end, we propose the joint application of LIME and attention-based techniques for a more thorough and balanced explanation that involves different components of the prediction process.

In terms of the hypotheses formulated in **Q2-1**, we arrived at **Hypothesis V**, according to which explanations can be generated by combining information from multiple sources listed in other hypotheses.

More specifically, our proposed solution utilizes the content of **Hypothesis I**, which implies that explanations related to existing stylometric features may be found by visualizing attention weights of different heads at different levels, and **Hypothesis IV**, which involves explanations based on potentially complex features extracted by means of input permutation.

In the course of that research, we disproved **Hypothesis III**, according to which the uppermost layer, the classifier itself, can provide sufficient explanations. We showed that the classifier merely uses the consistent differences between classes that are shaped and accumulated in the embeddings at the earlier stages.

The results of feature importance assessment and comparison of the two techniques enabled us to conclude that features extracted by then provide, at least to a certain extent, valid explanations that account for a large part of the model prediction. We can therefore use the feature classification discussed in Chapter 8 to answer the sub-question **Q2-1.1** by claiming that the model utilizes a large number of linguistic features on character, lexical, syntactic, and potentially semantic levels.

Many of them are considered by existing solutions, including frequencies of function words and numbers, the occurrence of punctuation and spelling errors, and vocabulary richness. Other features are not listed among traditional stylometric parameters but are nonetheless revealed in other solutions that use Attention for AA models (Boenninghoff et al., 2019). Among such features are proper nouns, special symbols, capitalization, and acronyms. A number of features that seem to be specific to the class of literary texts are not described in the works of which we are aware. Such features include reporting verbs, words used to refer to people, and quotes (the latter relates to the generic feature of punctuation counts, but we highlight the particular importance of quotes for the observed texts).

9.3 Future work

Hypothesis II was not taken into account based on the simplifying assumption we made when analyzing the attention weights. However, it is not refuted by any means, and we propose the analysis of dense layers inside BERT, as well as dimension-wise investigation of individual neurons in the query and key vectors, as potential directions for future research for more thorough interpretability of Transformer-based architectures.

For the ensemble architecture proposed by Peng et al. (2019), different ways of processing the output of all segments can be used. Instead of averaging the embeddings, some weighting techniques or additional dense layers can be employed to assign importance to embeddings of different fragments.

A variety of future directions for research concern different model architectures that may improve the performance or aid in generalizing the model to a broader range of tasks. For current research, we implemented the Transformer-based model for authorship identification with a widely used BERT model that has already been shown to perform successfully in related areas. However, we also note a variety of paths for further development of the model itself, that can even be possible within the adopted architecture.

Firstly, RoBERTa (Liu et al., 2020) was mentioned by (Fabien et al., 2020) and (Barlas and Stamatatos, 2021) as a desirable path for future experiments, and it has outperformed original BERT model in a number of applications, and Longformer (Beltagy et al., 2020), which is an extension of RoBERTa adapted for processing longer sequences, has shown better results for larger text chunks, which can be desirable in the attribution of literary works. Therefore, combining it with the ensemble technique may further improve the results.

Besides, a promising approach by (Katharopoulos et al. 2020) has not been used for language models yet, but the potential benefits are significant. Lastly, all of the previous research gave no account of how well the 3rd generation of GPT model could perform. GPT-2 (Radford et al., 2019) showed competitive results, though not the best ones on average, but GPT-3 (Brown et al., 2020) is already known for being capable of very diverse and complex linguistic tasks. Although access to GPT-3 itself is limited, GPT-J (Komatsuzaki, 2021) provides a publicly available analogue that achieved comparable results. Therefore, testing how well it can handle the AA tasks compared to BERT is of great interest.

The research of models with longer input coupled with feature extraction techniques could also help determine whether there exist some extremely-long-distance features that can be retrieved by the model and are useful for encoding the style of different authors. Such features are of great interest both for the study of attribution per se, and the philological research in general, as they may be related to the general composition and narrational structure of the text. However, extensive research will be required to analyze that features and bring them in correspondence with the existing formal methods of quantifying the style.

Another important direction of future work could be aimed at increasing the robustness of authorship attribution models. In our work, we checked the number of cases in which the predicted label was inverted for at least one important feature and discover that it happens in 50% of cases. The possibility of fooling the model by only changing one feature was also investigated with adversarial examples. Such ease of impeding the correct prediction goes in line with the observations of Boenninghoff et al. (2019) who noted that a model may rely heavily on a single feature and motivates the need for future research.

From a more general perspective, future research with an emphasis on causality is highly anticipated. Our suggested approach ensures a certain degree of objectivity by comparing the features produced by two techniques of different nature. Yet, an even more objective joint account of how changes in the input influence the way that input is processed inside the model can probably be achieved by incorporating gradient-based methods.

Ideally, such research could be performed in parallel with a detailed linguistic study of the texts to which the model is applied to arrive at a classification of ways in which different linguistic features may influence the style. Such classification is missing in the works concerning interpretable attribution since they typically concentrate on the set of features and not on the subcategories and their functions. This will facilitate the interpretability of explanations themselves since the problem of meta-interpretability is becoming an issue on its own. It is needed to keep in mind that, especially in sensitive areas, even the explanations of models' behavior cannot be trusted blindly. Their plausibility and truthfulness need to be assessed using methods grounded in the theory of a particular domain.

In conclusion, we would like to get back to the importance of designing the model itself and all its explanations with orientation for the end users, the linguist experts performing attributional studies. It is important to bear in mind that all findings provided by AI techniques need to be brought into correlation with real-life evidence in the end and be provided with plausible and truthful interpretations that can sensibly motivate the prediction. And even the explanation that is proven to be important and that is justified by linguistic analysis for a particular case still should not be trusted blindly. After all, citing Craig and Kinney (2009), "The results of computational linguistics are always matters of probability, not certainty. (...) After all, we are dealing with writers who are at liberty to imitate each other, to try new styles, and to write differently for a particular occasion or in a new genre"

Bibliography

1. Abnar, S., and Zuidema, W. (2020). Quantifying Attention Flow in Transformers. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (pp. 4190–4197). Association for Computational Linguistics.
2. Altakrori M, Cheung J, Fung B. (2021) The Topic Confusion Task: A Novel Evaluation Scenario for Authorship Attribution
3. Alvarez-Melis, D. and Jaakkola, T. (2017). A causal framework for explaining the predictions of black-box sequence-to-sequence models. 412-421. 10.18653/v1/D17-1042.
4. Argamon-Engelson, S., Koppel, M., and Avneri, G. (1998). Style-based text categorization: What newspaper am I reading? In Proceedings of AAAI Workshop on Learning for Text Categorization (pp. 1–4).
5. Argamon, S. Interpreting Burrows 'Delta: Geometric and Probabilistic Foundations. // Literary and Linguistic Computing 23 (2), — 2008. — 131-147.
6. Argamon, S., Konnel, M., Pennebaker, J., and Schier, J. (2007). Mining the Blogosphere: Age, gender and the varieties of self-expression. First Monday, 12.
7. Argamon, S., Whitelaw, C., Chase, P., Hota, S.R., Garg, N., and Levitan, S. (2007). Stylistic text classification using functional lexical features. Journal of the American Society for Information Science and Technology, 58(6), 802–822. arXiv preprint arXiv:1702.08608.
8. Baayen, R., van Halteren, H., and Tweedie, F. (1996). Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution. Literary and Linguistic Computing, 11(3), 121–131.
9. Bagnall, D. (2015) Author identification using multi-headed recurrent neural networks. In: Working Notes of CLEF 2015—Conference and Labs of the Evaluation forum
10. Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. ArXiv. 1409.
11. Barlas, G. and Stamatatos E. (2020). Cross-Domain Authorship Attribution Using Pre-trained Language Models. In Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, volume 583, pages 255–266. Springer International Publishing, Cham. Series Title: IFIP Advances in Information and Communication Technology.
12. Barlas, G., Stamatatos, E. (2021) A transfer learning approach to cross-domain authorship attribution. https://doi-org.proxy.library.uu.nl/10.1007/s12530-021-09377-2
13. Beltagy, I., Peters, M.E., Cohan, A. (2020). Longformer: The long-document transformer. ArXiv abs/2004.05150 (2020)
14. Benedetto, D., Caglioti, E., and Loreto, V. (2002). Language trees and zipping. Physical Review Letters, 88(4), 048702.
15. Bengio, Y., Frasconi, P., Schmidhuber, J. (2003). Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. A Field Guide to Dynamical Recurrent Neural Networks.
16. Bischoff, S., Deckers, N., Schliebs, M., Thies, B., Hagen, M., Stamatatos, E., Stein,B., Potthast, M.: The importance of suppressing domain style in authorship analysis. CoRR abs/2005.14714 (2020), URL https://arxiv.org/abs/2005.14714
17. Boenninghoff, B., Hessler, S., Kolossa, D., Nickel, R. (2019). Explainable Authorship Verification in Social Media via Attention-based Similarity Learning. 36-45. 10.1109/BigData47090.2019.9005650.
18. Brown, T. B., Mann, B. P., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krüger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E.J., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D. (2020). Language models are few-shot learners. ArXiv abs/2005.14165
19. Burrows, J. (1987). Word patterns and story shapes: The statistical analysis of narrative style. Literary and Linguistic Computing, 2, 61–70.

20. Burrows, J. (1989). 'An ocean where each kind. . . ': Statistical analysis and some major determinants of literary style. Computers and the Humanities, vol. 23, no. 4–5, pp. 309–21, 1989.
21. Burrows, J. (1992). Not unless you ask nicely: The interpretative nexus between analysis and information. Literary and Linguistic Computing, 7(2), 91–109.
22. Burrows, J. (2002). Delta: A measure of stylistic difference and a guide to likely 45 authorship. Literary and Linguistic Computing, 17(3), 267-287.
23. Burrows J. (2007) All the way through: testing for authorship in different frequency strata. // Literary and Linguistic Computing, 22(1), — 2007. — pp. 27–48
24. Campbell L. (1867) The Sophisties and Polilicus of Plato — Oxford : Clarendon, 1867. — 170 p.
25. Carter, R. English Grammar Today: The Cambridge A-Z Grammar of English, 2016.
26. Chefer, H., Gur, S., and Wolf, L.. (2020). Transformer Interpretability Beyond Attention Visualization.
27. Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2013) One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.
28. Cheng, J., Dong, L., Lapata, M. (2016). Long Short-Term Memory-Networks for Machine Reading.
29. Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.
30. Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014) Learning phrase representations using rnn encoderdecoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.
31. Choi, E., Bahadori, M., Kulas, J., Schuetz, A., Stewart, W., and Sun, J.. (2016). RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism. In Advances in Neural Information Processing Systems, pages 3504–3512.
32. Chung, J., Gülçehre, C., Cho, K., and Bengio, Y. (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/ 1412.3555, 2014.
33. Cilibrasi, R., and Vitanyi, P.M.B. (2005). Clustering by compression. IEEE Transactions on Information Theory, 51(4), 1523–1545.
34. Clark, K., Khandelwal, U., Levy, O., and Manning, C. (2019). What Does BERT Look At? An Analysis of BERT's Attention.
35. Craig H., and Kinney A.F. (2009). Shakespeare, Computers, and the Mystery of Authorship, Cambridge, Cambridge University Press.
36. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). TransformerXL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978– 2988, Florence, Italy, July 2019. Association for Computational Linguistics.
37. Danilevsky, M., Qian, K., Aharonov, R., Katsis, Y., Kawas, B., and Sen, P. (2020). A Survey of the State of Explainable AI for Natural Language Processing. arXiv preprint arXiv:2010.00711, 2020.
38. de Vel, O., Anderson, A., Corney, M., and Mohay, G. (2001). Mining e-mail content for author identification forensics. SIGMOD Record, 30(4), 55–64.
39. Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. 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), pp. 4171– 4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
40. Doshi-Velez, F., and Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning.

41. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.
42. Eder, M. and Rybicki, J. Do birds of a feather really flock together, or how to choose training samples for authorship attribution. // Literary and Linguistic Computing 28 (2), —2013. — pp. 229-236.
43. Eder, M., Rybicki, J. and Kestemont, M. (2016). Stylometry with R: a package for computational text analysis. R Journal 8(1): 107-121. <https://journal.r-project.org/archive/2016/RJ-2016-007/index.html>
44. Eder. M. (2012) Computational stylistics and biblical translation: how reliable can a dendrogram be? The Translator and the Computer. — WSF Press, Wroclaw, 2012. — pp. 155–170
45. Ehsan, U., Tambwekar, P., Chan, L., Harrison, B., and Riedl, M.. (2019). Automated Rationale Generation: A Technique for Explainable AI and its Effects on Human Perceptions.
46. Evert St., Proisl Th., Jannidis F., Reger Is., Pielström St., Schöch Chr., Vitt. Th. Understanding and explaining Delta measures for authorship attribution // Digital Scholarship in the Humanities, Vol. 32, Issue 2, — 2017. — pp. 114–116
47. Fabien, M., Villatoro-Tello, E., Motlicek, P., and Parida, S. (2020). BertAA: BERT fine-tuning for Authorship Attribution. In Proceedings of the 17th International Conference on Natural Language Processing (ICON) (pp. 127–137). NLP Association of India (NLPAI).
48. Feng, S., Wallace, E., Grissom II, A., Iyyer, M., Rodriguez, P., and Boyd-Graber, J. (2018). Pathologies of Neural Models Make Interpretations Difficult. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (pp. 3719–3728). Association for Computational Linguistics.
49. Fix, E., Hodges, J. L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties (Report). USAF School of Aviation Medicine, Randolph Field, Texas
50. Forsyth, R. S. and Holmes, D. I. (1996) Feature-finding for text classification. // Literary and Linguistic Computing, 11. — 1996. — pp. 163–74
51. Futrzynski, R. Author classification as pre-training for pairwise authorship verification, in: G. Faggioli, N. Ferro, A. Joly, M. Maistro, F. Piroi (Eds.) (2021). CLEF 2021 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2021.
52. Gamon, M. (2004). Linguistic correlates of style: Authorship classifica-tion with deep linguistic analysis features. In Proceedings of the 20th International Conference on Computational Linguistics (pp. 611–617). Morristown, NJ: Association for Computational Linguistics.
53. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y. (2017). Convolutional Sequence to Sequence Learning.
54. Ghaeini, R., Fern, X., and Tadepalli, P. (2018). Interpreting Recurrent and Attention-Based Neural Models: a Case Study on Natural Language Inference. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (pp. "4952–4957",). Association for Computational Linguistics.
55. Glaudes P., Guglielmi F., Mayaux C., Marusenko M. A., Kuralesina E. N., Miretina M. S., Nikitina E. Ya., Solovieva M. V., Khutoretskaya O. A., Kondyurine I. A. (2019). Jules Barbey d'Aurevilly and Newspaper Corpus: problems of attribution (Part II) eISSN: 0202-2502
56. Goldstein-Stewart J, Winder R, Sabin RE (2009) Person identification from text and speech genre samples. https://aclanthology.org/E09-1039.pdf
57. Graves A. (2013) Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
58. Halvani, O., Graner,L. (2018) Cross-domain authorship attribution based on compression: Notebook for PAN at CLEF 2018. In: Cappellato, L., Ferro, N., Nie, J., Soulier, L. (eds.) Working Notes of CLEF 2018 Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018, CEUR Workshop Proceedings, vol. 2125, CEUR-WS.org (2018), URL http://ceur-ws.org/Vol-2125/paper_90.pdf

59. Hill, F., Bordes, A., Chopra, S., and Weston, J. (2015) The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.
60. Hirst, G. and Feiguina, O. (2007). Bigrams of syntactic labels for authorship discrimination of short texts. Literary and Linguistic Computing, 22(4), 405–417.
61. Hochreiter, S. and Schmidhuber, J. (1997) Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
62. Holmes, D. I., The Evolution of Stylometry in Humanities Scholarship, Literary and Linguistic Computing 13 (1998) 111–117.
63. Holmes, H. (1994). Authorship Attribution, Computers and Humanities, p. 87
64. Hoover D. L. Testing Burrows's Delta // Literary and Linguistic Computing, Vol. 19, No. 4. — 2004. —pp. 453–475
65. Hoover, D. L. (2003). Another Perspective on Vocabulary Richness. Computers and the Humanities. 37, 151-178.
66. Hoover, D. L. Delta, Delta Prime, and Modern American Poetry: Authorship Attribution Theory and Method. // Proceedings of the 2005 ALLC/ACH conference. — 2005.
67. Hoover, J. F. (2004). Delta Prime? Literary and Linguistic Computing, 19(4), 477-495.
68. Houvardas, J., and Stamatatos, E. (2006). N-gram feature selection for authorship identification. In Proceedings of the 12th International Conference on Artificial Intelligence: Methodology, Systems, Applications (pp. 77–86). Berlin, Germany: Springer.
69. Howard, J., and Ruder, S.. (2018). Universal Language Model Fine-tuning for Text Classification.
70. Hulley, K. K. (1944). Principles of Textual Criticism Known to St. Jerome. Harvard Studies in Classical Philology, 55, 87–109. https://doi.org/10.2307/310878
71. Jain, S. and Byron C. Wallace, B. C. (2019). Attention is not Explanation. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Minneapolis, Minnesota. Association for Computational Linguistics.
72. Jannidis, F., Pielstrom, S., Schoch, Ch. and Vitt, Th. Improving Burrows' Delta: An empirical evaluation of text distance measures. // Digital Humanities 2015: Conference Abstracts. — 2015.
73. Jawahar G, Sagot B, Seddah D (2019). What does BERT learn about the structure of language? In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, pp 3651–3657
74. Johnson, A., Pollard, T., Shen, L., Lehman, L.w., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L., and Mark, R. (2016). MIMIC-III, a freely accessible critical care database. Scientific Data, 3, 160035.
75. Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L.. (2017). TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension.
76. Juola, P. (1997). What can we do with small corpora? Document categorization via cross-entropy," in Proceedings of an Interdisciplinary Workshop on Similarity and Categorization, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, UK, 1997.
77. Juola, P. (2003). The time course of language change. Computers and the Humanities, vol. 37, no. 1, pp. 77–96, 2003.
78. Juola, P. (2006). Authorship attribution. Foundations and Trends in Information Retrieval, 1(3).
79. Katharopoulos, A., Vyas, A., Pappas, N., Fleuret, F. (2020). Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. Proceedings of the 37th International Conference on Machine Learning, in Proceedings of Machine Learning Research 119:5156-5165 Available from https:// proceedings.mlr.press/v119/katharopoulos20a.html.
80. Kenny, A. (1981) Some observations on the stylometry of the Pauline epistles // Actes du Congrès international informatique et sciences humaines. —L.A.S.L.A. Liege, Belgium, 1981. — pp. 510–512
81. Kestemont M, Tschuggnall M, Stamatatos E, Daelemans W, Specht G., Stein B., Potthast M. (2018). Overview of the Author Identification Task at PAN-2018: Cross-domain Authorship Attribution and Style Change Detection. In: Cappellato L, Ferro N, Nie Y, Soulier L (eds) Working

Notes Papers of the CLEF 2018 Evaluation Labs, *CEUR Workshop Proceedings*, vol. 2125. CEUR-WS.org . http:// ceur-ws.org/ Vol-2125/

82. Kestemont, M., Manjavacas, E., Markov, I., Bevendorff, J., Wiegmann, M., Stamatatos, E., Potthast, M., and Stein, B. (2020). Overview of the Cross-Domain Authorship Verification Task at PAN 2020. In B. De Carolis, C. Gena, A. Lieto, S. Rossi, and A. Sciutti (Eds.), *cAESAR 2020 - Proceedings of the Workshop on Adapted Interaction with Social Robots* (Vol. 2696). (CEUR Workshop Proceedings). CEUR-WS.
83. Kestemont, M., Stamatatos, E., Manjavacas, E., Daelemans, W., Potthast, M., Stein, B. (2019). Overview of the cross-domain authorship attribution task at PAN 2019. In: Cappellato L, Ferro N, Losada DE, Müller H (eds) Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum, *CEUR Workshop Proceedings*, vol. 2380. CEUR-WS.org
84. Kestemont, M., Stover, J.A., Koppel, M., Karsdorp, F., Daelemans, W. (2016) Authenticating the writings of julius caesar. Expert Systems with Applications 63, 86–96 (2016), https://doi.org/10.1016/ j.eswa.2016.06.029
85. Kestemont, M., Manjavacas, E., Markov, I., Bevendorff, J., Wiegmann, M., Stamatatos, E., Stein, B., Potthast, M. (2021). Overview of the Cross-Domain Authorship Verification Task at PAN 2021.
86. Khosmood, F., and Levinson, R. (2006). Toward unification of source attribution processes and techniques. In Proceedings of the 5th International Conference on Machine Learning and Cybernetics (pp. 4551–4556). Washington, DC: IEEE.
87. Kim, Y., Denton, C., Hoang, L., and Rush, A. (2017). Structured Attention Networks.
88. Kitaev, N., Kaiser, Ł., and Levskaya, A. (2020). Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451, 2020.
89. Kjell, B. (1994). Discrimination of authorship using visualization. Information Processing and Management, 30(1), 141–150.
90. Klimt., B. and Yang., Y. (2004). The Enron Corpus: A New Dataset for Email Classification Research. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Jean-Franc¸Lois Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004*, volume 3201, pages 217–226. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title: Lecture Notes in Computer Science.
91. Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese Neural Networks for One-shot Image Recognition.
92. Komatsuzaki A. (2021) GPT-J-6B: 6B JAX-Based Transformer https:// arankomatsuzaki.wordpress.com/2021/06/04/gpt-j/
93. Koppel M., Schler J., Argamon S., Winter Y. (2012). The “Fundamentals Problem” of Authorship Attribution. English Studies, 93(3), 284-291.
94. Koppel, M., and Schler, J. (2003). Exploiting stylistic idiosyncrasies for authorship attribution. In Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis (pp. 69–72).
95. Koppel, M., Akiva, N., and Dagan, I. (2006). Feature instability as a criterion for selecting potential style markers. Journal of the American Society for Information Science and Technology, 57(11), 1519–1525.
96. Koppel, M., Schler, J. and Argamon, S. (2009) Computational Methods in Authorship Attribution. // Journal of the American Society for Information Science and Technology 60 (1), —2009. — 9-26.
97. Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. (2019). Revealing the Dark Secrets of BERT. In *EMNLP/IJCNLP*.
98. Kuchaiev, O. and Ginsburg, B. (2017) Factorization tricks for LSTM networks. *arXiv preprint arXiv:1703.10722*, 2017.

99. Kukushkina, O., Polikarpov, A., and Khmelev, D. (2001). Using literal and grammatical statistics for authorship attribution. Problems of Information Transmission, 37(2), 172–184.
100. Lample, G., Sablayrolles, A., Ranzato, M. A., Denoyer, L., and Jegou, H. (2019) Large memory layers with product keys. In Wallach, H., Larochelle, H., Beygelzimer, A., dA ́lche ́-Buc,F.,Fox,E.,andGarnett,R.(eds.),*Advances in Neural Information Processing Systems 32*, pp. 8546– 8557. Curran Associates, Inc., 2019.
101. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.
102. Lei, T., Barzilay, R., and Jaakkola, T. (2016). Rationalizing Neural Predictions. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (pp. 107–117). Association for Computational Linguistics.
103. Li, J., Monroe W., and Jurafsky, D. (2016). Understanding neural networks through representation erasure. arXiv preprint arXiv:1612.08220.
104. Li, J., Zheng, R., and Chen, H. (2006). From fingerprint to writeprint. Communications of the ACM, 49(4), 76–82.
105. Li, M. and Vitanyi, P. (1997). An Introduction to Kolmogorov Complexity and Its Applications. Graduate Texts in Computer Science, New York: Springer, second ed., 1997.
106. Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer., N. (2018) Generating wikipedia by summarizing long sequences. *ICLR*, 2018.
107. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2020). RoBERTa: A robustly optimized BERT pretraining approach, 2020.
108. Logeswaran, L. and Lee, H. (2018). An efficient framework for learning sentence representations.
109. Loshchilov, I., and Hutter, F. (2017). Decoupled Weight Decay Regularization.
110. Luong, M., Pham, H., Manning, C. (2015). Effective Approaches to Attention-based Neural Machine Translation. 10.18653/v1/D15-1166.
111. Lutoslawski, W. (1898). Principes de stylométrie appliqués à la chronologie des œuvres de Platon // Revue des Études Grecques. —1898. —pp. 61–81
112. Martins, A. and Astudillo, R. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623.
113. Marton, Y., Wu, N., and Hellerstein, L. (2005). On compression-based text classification. In Proceedings of the European Conference on Information Retrieval (pp. 300–314).
114. Marusenko M. (1990). Attribution of anonymous and pseudonymous literary works by means of pattern recognition [Марусенко М. А. Атрибуция анонимных и псевдонимных литературных произведений методами теории распознавания образов. — Л.: Изд-во ЛГУ, 1990.]
115. McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in Translation: Contextualized Word Vectors.
116. Mead, A. (1992). Review of the Development of Multidimensional Scaling Methods. Journal of the Royal Statistical Society. Series D (The Statistician). 41 (1): 27–39.
117. Mendenhall T. (1887). The characteristic curves of composition. Science, vol. IX, pp. 237–249, 1887.
118. Merriam, T (1994). Letter Frequency as a Discriminator of Authors. Notes & Queries, 239, 1994, p. 467-469.
119. Merriam, T (1998). Heterogeneous Authorship in Early Shakespeare and the Problem of Henry V. Literary and Linguistic Computing, 13, 1998, p. 15-28.
120. Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. 353-355. 10.18653/v1/W18-5446.
121. Michel, P., Levy, O., and Neubig, G. (2019). Are sixteen heads really better than one? In Wallach, H., Larochelle, H., Beygelzimer, A., d 'Alche ́-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 14014–14024. Curran Associates, Inc., 2019.
122. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In: Conference on Advances in Neural Information

Processing Systems. Distributed Representations of Words and Phrases and Their Compositionality. 3111-3119

123. Miller, G. A., Beckwith, R., Fellbaum, C. D., Gross, D., Miller, K. (1990). WordNet: An online lexical database. Int. J. Lexicograph. 3, 4, pp. 235–244.
124. Morton, A. Q., and McLeman, J. (1966). Paul, the man and the myth: A study in the authorship of Greek prose. —New York: Harper & Row, 1966.
125. Mosteller, F. and Wallace, D. L. (1964) Inference and Disputed Authorship: The Federalist Papers. — New York: Springer, 1964
126. Mullenbach, J., Wiegreffe, S., Duke, J., Sun, J., and Eisenstein, J. (2018). Explainable Prediction of Medical Codes from Clinical Text. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (pp. "1101–1111",). Association for Computational Linguistics.
127. Neal, T., Sundararajan, K., Fatima, A., Yan, Y., Xiang, Y., and Woodard, D. (2017). Surveying Stylometry Techniques and Applications. ACM Computing Surveys, 50, 1-36.
128. Ordoñez, J., Soto, R.A., and Chen, B.Y. (2020). Will Longformers PAN Out for Authorship Verification? Notebook for PAN at CLEF 2020. CLEF.
129. o'Riedl, M. (2019). Human-centered artificial intelligence and machine learning. Human Behavior and Emerging Technologies, 1(1):33–36.
130. Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernandez, R. (2016) The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.
131. Papineni, K., Roukos, S., Ward, T., Zhu, W. J. (2002). BLEU: a method for automatic evaluation of machine translation (PDF). ACL-2002: 40th Annual meeting of the Association for Computational Linguistics. pp. 311–318. CiteSeerX 10.1.1.19.9416
132. Pappas, N., and Popescu-Belis, A. (2016). Human versus machine attention in document classification: A dataset with crowdsourced annotations. In Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media, pages 94–100.
133. Kharya, P. and Alvi, A. (2021) Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, the World's Largest and Most Powerful Generative Language Model https://developer.nvidia.com/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/
134. Paulus, R., Xiong, C., and Socher, R. (2017) A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.
135. Peters, B., Niculae, V., and Martins, A. (2018). Interpretable Structure Induction via Sparse Attention. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP (pp. 365–367). Association for Computational Linguistics.
136. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. (2018). Deep Contextualized Word Representations. 2227-2237. 10.18653/v1/N18-1202.
137. Radford, A., Narasimhan, K., Salimans, T., , and Sutskever, I. (2018). Improving language understanding by generative pretraining. In *OpenAI report*, 2018.
138. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.
139. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P.. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, 21:1–67.
140. Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text.
141. Ribeiro, M., Singh, S., and Guestrin, C. (2016). "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 (pp. 1135–1144). Association for Computing Machinery.

142. Riedl, M. O. Human-centered artificial intelligence and machine learning. Human Behavior and Emerging Technologies, 1(1):33–36. https://doi.org/10.1002/hbe2.117
143. Rodionova E. (2007) Informative parameters selection for the attribution of verse plays by Molière [Родионова Е.С. Отбор информативных параметров при атрибуции стихотворных пьес Мольера // Материалы XXXVI Международной филологической конференции (12 – 17 марта 2007 г.). – СПб : Филол. фак. С.-Петерб. гос. ун-та, 2007. – Вып. 10 : Прикладная и математическая лингвистика / под ред. Т. Г. Скребцовой. С. 67–74]
144. Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A Primer in BERTology: What We Know About How BERT Works. Transactions of the Association for Computational Linguistics, 8, 842–866.
145. Ross, A., Hughes, M., and Doshi-Velez, F.. (2017). Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, pages 2662–2670. AAAI Press.
146. Rosset, C. (2020). Turing-NLG: A 17-billion-parameter language model by microsoft. *Microsoft Research Blog*, 2:13.
147. Rudin, C. (2018). Please stop explaining black box models for high stakes decisions. *arXiv preprint arXiv:1811.10154.*
148. Rybicki, J. and Eder, M. Deeper Delta across genres and languages: do we really need the most frequent words? // Literary and Linguistic Computing 26 (3), — 2011. — pp. 315-321.
149. Sapkota, U., Solorio, T., Montes, M., Bethard, S., and Rosso, P. (2014). Cross-Topic Authorship Attribution: Will Out-Of-Topic Data Help?. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers (pp. 1228–1237). Dublin City University and Association for Computational Linguistics.
150. Sari, Y., Stevenson, M., and Vlachos, A. (2018). Topic or Style? Exploring the Most Useful Features for Authorship Attribution, Proceedings of the 27th International Conference on Computational Linguistics. pages 343–353 Santa Fe, New Mexico, USA, August 20-26, 2018. https://aclanthology.org/C18-1029.pdf
151. Schuster, M., and Nakajima, K. (2012). Japanese and Korean voice search. In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 5149-5152).
152. Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Computing Surveys, 34(1), 1–47.
153. Serrano, S., and Smith, N. (2019). Is Attention Interpretable? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (pp. 2931–2951). Association for Computational Linguistics.
154. Seroussi, Y., Zukerman, I., and Bohnert, F. (2014). Authorship Attribution with Topic Models. Computational Linguistics, 40(2), 269–310.
155. Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, Manaal Faruqui. (2019). Attention Interpretability Across NLP Tasks In ICLR 2020 Conference Blind Submission
156. Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. (2019). Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.
157. Simpson, E. H. (1949). Measurement of diversity. Nature, vol. 163, p. 688, 1949.
158. Smith, P. and Aldridge W. Improving Authorship Attribution. Optimizing Burrows 'Delta Method. // Journal of quantitative linguistics 18(1), — 2017. — pp. 63-88.
159. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts., C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.
160. Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. MASS: Masked sequence to sequence pre-training for language generation. In Chaudhuri, K. and Salakhutdinov, R. (eds.) (2019). *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5926–5936, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

161. Stamatatos, E, Barlas G. (2020) Authorship Attribution Using Pre-trained Language Models https://link-springer-com/chapter/10.1007%2F978-3-030-49161-1_22
162. Stamatatos, E, Rangel, F., Tschuggnall, M., Stein, B., Kestemont, M., Rosso, P., Potthast, M. (2018) Overview of pan 2018. In: International Conference of the Cross-Language Evaluation Forum for European Languages. Springer, pp 267–285 http://ceur-ws.org/Vol-2125/invited_paper_2.pdf
163. Stamatatos, E. (2009). A survey of modern authorship attribution methods. Journal of the American Society for Information Science and Technology, vol 60, no 3, pp 538–556 https://doi-org.proxy.library.uu.nl/10.1002/asi.21001
164. Stamatatos, E. (2013) On the Robustness of Authorship Attribution Based on Character N-gram Features https://brooklynworks.brooklaw.edu/cgi/viewcontent.cgi?article=1048&context=jlp
165. Stamatatos, E. (2017) Masking topic-related information to enhance authorship attribution https://asistdl-onlinelibrary-wiley-com.proxy.library.uu.nl/doi/10.1002/asi.23968
166. Stamatatos, E., Fakotakis, N., and Kokkinakis, G. (2001). Computer-based authorship attribution without lexical measures. Computers and the Humanities, 35(2), 193–214.
167. Sukhbaatar, S., Grave, E., Bojanowski, P., and Joulin, A. (2019). Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 331–335, Florence, Italy, July 2019. Association for Computational Linguistics.
168. Suman, C, Raj, A, Saha, S, and Bhattacharyya, P. (2021). Authorship Attribution of Microtext Using Capsule Networks," in IEEE Transactions on Computational Social Systems, doi: 10.1109/TCSS.2021.3067736
169. Sun, C., Qiu, X., Xu, Y., and Huang, X. (2020). How to Fine-Tune BERT for Text Classification?
170. Sutskever, I., Vinyals, O., and Quoc V. Le. (2014) Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104– 3112, 2014.
171. Tanaka-Ishii, K., Aihara S. (2015). Computational Constancy Measures of Texts — Yule's K and Rényi's Entropy. Computational Linguistics 2015; 41 (3): 481–502. doi: https://doi.org/10.1162/COLI_a_00228
172. Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2019). Generating Token-Level Explanations for Natural Language Inference. 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) (pp. 963–969). Association for Computational Linguistics.
173. Tyo, J., Dhingra, B., Lipton, Z., Siamese Bert for authorship verification, in: G. Faggioli, N. Ferro, A. Joly, M. Maistro, F. Piroi (Eds.), CLEF 2021 Labs and Workshops, Notebook Papers, CEUR-WS.org, 2021.
174. van Halteren, H., Baayen, R. H., Tweedie, F., Haverkort, M., and Neijt, A. (2005). New machine learning methods demonstrate the existence of a human stylome. // Journal of Quantitative Linguistics, vol. 12, no. 1. — 2005. — pp. 65–77
175. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *NIPS*, 2017.
176. Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf (2019) DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter
177. Vig, J. (2019). A Multiscale Visualization of Attention in the Transformer Model. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 37–42). Association for Computational Linguistics.
178. Vig, J. and Belinkov, Y. (2019) Analyzing the structure of attention in a transformer language model. In Proc. of BlackBoxNLP, 2019.
179. Wiegreffe, S., and Pinter, Y. (2019). Attention is not not Explanation. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 11–20). Association for Computational Linguistics.

180. Williams, A., Nangia, N., and Bowman, S. (2018). A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. 1112-1122. 10.18653/v1/N18-1101.
181. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-the-Art Natural Language Processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (pp. 38–45). Association for Computational Linguistics.
182. Xie, Q., Ma, X., Dai, Z., and Hovy, E. (2017). An Interpretable Knowledge Transfer Model for Knowledge Base Completion. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. "950–962",). Association for Computational Linguistics.
183. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y.. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.
184. Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/ 1906.08237, 2019.
185. Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. (2018). Deriving machine attention from human rationales. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1903–1913.
186. Yule, G. (1938). On sentence-length as a statistical characteristic of style in prose, with application to two cases of disputed authorship. Biometrika, 30, 363–390.
187. Yule, G. (1944). The Statistical Study of Literary Vocabulary. Cambridge University Press.
188. Zachary C. Lipton. (2016). The mythos of model interpretability. *arXiv preprint arXiv:1606.03490.*
189. Zafrir, O., Boudoukh, G., Izsak, P., and Wasserblat, M. (2019). Q8BERT: quantized 8bit BERT. *CoRR*, abs/1910.06188, 2019.
190. Zein Shaheen, Gerhard Wohlgenannt, Erwin Filtz. (2021) Large Scale Legal Text Classification Using Transformer Models arXiv:2010.12871
191. Zellers, R., Bisk, Y., Schwartz, R., and Choi, Y. (2018). SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP).*
192. Zhao, Y., and Zobel, J. (2005). Effective and scalable authorship attribution using function words. In Proceedings of the 2nd Asia Information Retrieval Symposium. Berlin, Germany: Springer.
193. Zheng, R., Li, J., Chen, H., and Huang, Z. (2006). A framework for authorship identification of online messages: Writing style features and classification techniques. Journal of the American Society of Information Science
194. Zipf, G. (1932). Selected studies of the principle of relative frequency in language. Cambridge, MA: Harvard University Press.

Appendix

A. Description of all CLS-related attention

Layer	Head	Direction	Important Tokens	Types of Important Tokens	Strength of attention	Scope	Coverage (one text or both)
1	1	To CLS	Names, subword tokens (all), topical nouns	Content words, subword tokens	Moderately strong	Global	Both
1	2	To CLS	Names, subword tokens (all), topical nouns, narrational verbs	Content words, subword tokens	Strong	Global	Both
1	5	To CLS	Names, subword tokens (initial)	Content words, subword tokens	Moderately strong	Global	Both
1	7	Between tokens (between non-narrational verbs)	Non-narrational verbs	Content words, subword tokens	Moderately strong	Large	Both
1	9	Between tokens (narrational verbs and full/short form variations)	Narrational verbs, full/short form variations	Content words, special function words	Moderately strong	Local	Both
1	9	To CLS	Names, words co-referring to named persons	Content words	Moderately strong	Global	Both
2	7	To CLS	Subword tokens (non-initial), prepositions	Function words, subword tokens	Moderately strong	Global	Both
2	9	To CLS	Names, stylistically variable words, in-word mistakes	Content words, subword tokens, function words, mistakes	Moderately strong	Global	Both
2	10	To CLS	Names, stylistically variable words, diatopic variations	Content words, subword tokens, function words, mistakes	Moderately strong	Global	Both
3	1	To CLS	Names, stylistically variable set phrases	Content words, set phrases	Moderately strong	Global	Both
3	4	From CLS	Names, verbs related to names, quotes	Content words, punctuation	Moderately strong	Global	Both
3	5	To CLS	Topical nouns, subword tokens (all), full stops, quotes	Content words, subword tokens, punctuation	Moderately strong	Global	Both
3	5	From CLS	Uniform attention	Uniform attention	Weak	Global	Both
3	5	Between tokens (words to punctuation)	Full stops, commas, quotes, question marks, other punctuation	Punctuation	Moderately strong	Local	Both
3	6	Between tokens (identity relation)	Subword tokens, prepositions, wh-words, pronouns, in-word mistakes	Function words, subword tokens, mistakes	Strong	Large	Both
3	9	To CLS	Infrequent narrational verbs, stylistically variable set phrases, topical nouns, other verbs	Content words, set phrases	Strong	Large	Both
3	10	Between subword tokens	Names, in-word mistakes, abbreviations	Subword tokens	Moderately strong	Neighbor	Both
3	12	From CLS	Narrational verbs, punctuation	Content words, punctuation	Moderately strong	Global	Both
4	1	Between tokens (narrational verbs and quotes)	Narrational verbs, quotes	Content words, punctuation	Moderately strong	Local	Both
4	2	From CLS	Names, narrational verbs, stylistically variable words, diatopic variations, full/short form variations, infrequent words, various mistakes, obscene words, numbers	Content words, special function words, mistakes, numbers	Moderately strong	Large	First
4	2	Between tokens (punctuation)	Punctuation	Punctuation	Moderately strong	Global, but stronger locally	Both

4	3	From CLS	Names, narrational verbs, stylistically variable words, diatopic variations, full/short form variations, infrequent words, various mistakes, obscene words, numbers	Content words, special function words, mistakes, numbers	Moderately strong	Large	Second
4	3	Between tokens ("important" words attended by other tokens)	Names, narrational verbs, stylistically variable words, diatopic variations, full/short form variations, infrequent words, various mistakes, obscene words, numbers	Content words, special function words, mistakes, numbers	Moderately strong	Medium	Both
4	4	From CLS	Names, topical nouns	Content words	Moderately strong	Global	Both
4	8	Between tokens (identity relation)	Subword tokens, prepositions, wh-words, pronouns, in-word mistakes	Function words, subword tokens, mistakes	Strong	Large	Both
4	9	From CLS	Uniform attention (end of Text 1)	Uniform attention	Weak	Local	First
4	12	To CLS	Names, punctuation	Content words, punctuation	Strong	Global	Both
5	2	From CLS	Names, narrational verbs, stylistically variable words, diatopic variations, full/short form variations, infrequent words, various mistakes, obscene words, numbers	Content words, special function words, mistakes, numbers	Moderately strong	Large	Second
5	5	From CLS	Names	Content words	Moderately strong	Large	Second
5	5	Between tokens (full/short form variations)	Full/short form variations	Function words	Moderately strong	Large	Both
5	7	From CLS	Names, topical nouns	Content words	Moderately strong	Large	First
5	7	Between tokens (words in utterance and quotes)	Quotes	Punctuation	Moderately strong	Local	Both
5	10	From CLS	Names	Content words	Moderately strong	Large	Second
5	11	From CLS	Pronouns, conjunctions, full/short form variations, punctuation	Function words, punctuation	Moderately strong, instance-specific	Large	Second
5	12	From CLS	Punctuation	Punctuation	Weak	Large	Second
6	6	Between tokens (punctuation and function words)	Pronouns, auxiliary verbs, punctuation	Function words, punctuation	Moderately strong	Medium	Both
6	7	From CLS	Names, numbers	Content words, numbers	Weak	Large	Second
6	9	From CLS	Names	Content words	Weak	Large	Second
6	10	From CLS	Names	Content words	Strong, instance-specific	Large	Second
7	1	From CLS	Names	Content words	Moderately strong	Large	First
7	7	Between tokens (conditional clauses, appositive clauses and direct speech)	Conjunctions, wh-words, modal verbs, narrational verbs, quotes, commas, colons	Content words, function words, punctuation	Strong	Local	Both
7	8	Between subword tokens	Names, in-word mistakes, abbreviations	Subword tokens	Strong	Neighbor	Both
7	9	From CLS	Names, narrational verbs, stylistically variable words, diatopic variations, full/short form variations, infrequent words, various mistakes, obscene words, numbers, punctuation	Content words, special function words, mistakes, numbers, punctuation	Moderately strong	Large	First
7	11	Between tokens (set phrases, objects with attributes, compound words with hyphens)	Pronouns, verbs, hyphens, other words	Set phrases, compound words	Very strong	Neighbor	Both
7	12	From CLS	Names	Content words	Moderately strong	Large	Second

8	1	Between tokens (punctuation and function words)	Casing mistakes	Mistakes	Strong, instance-specific	Local	Both
8	2	Between tokens (punctuation and function words)	Ellipsis (three dots), conjunctions, full/short form variations, pronouns, names	Content words, function words, punctuation	Strong, instance-specific	Local	Both
8	3	Between tokens (narrational verbs and quotes)	Narrational verbs, quotes, wh-words	Content words, function words, punctuation	Strong	Large	Both
8	4	From CLS	Names, narrational verbs	Content words	Moderately strong	Large	Second
8	6	From CLS	Names	Content words	Strong	Large	Second
8	6	Between tokens ("important" words attended by other tokens)	Names, narrational verbs, stylistically variable words, diatopic variations, full/short form variations, infrequent words, various mistakes, obscene words, numbers, punctuation	Content words, special function words, mistakes, numbers, punctuation	Moderately strong	Medium	Both
8	11	From CLS	Names	Content words	Moderately strong	Large	Second
8	12	From CLS	Quotes	Punctuation	Moderately strong, instance-specific	Global	Both
9	3	Between tokens (punctuation and function words)	Ellipsis (three dots), modal verbs, various function words, quotes	Function words, punctuation	Moderately strong	Large	Both
9	4	From CLS	Names, punctuation, pronouns	Content words, function words, punctuation	Strong	Global	Both, but mainly second
9	6	From CLS	Names	Content words	Moderately strong	Global	Both
9	7	Between tokens (punctuation and function words to verbs)	Auxiliary verbs, frequent verbs, narrational verbs, rare verbs, punctuation	Content words, function words, punctuation	Moderately strong	Large	Both
9	8	From CLS	Conjunctions, prepositions, adverbs	Function words	Moderately strong	Large	Second
9	9	From CLS	Names	Content words	Strong, instance-specific	Global	Both
9	11	From CLS	Names, punctuation	Content words, punctuation	Strong	Large	Second
9	11	Between tokens (verbs to punctuation and function words)	Full stops, commas, question marks, other punctuation, verbs, prepositions	Content words, function words, punctuation	Moderately strong	Local	Both
9	12	From CLS	Names	Content words	Moderately strong	Global	Both
10	1	From CLS	Quotes, full stops	Punctuation	Moderately strong	Global	Both
10	2	From CLS	Names	Content words	Moderately strong	Global	Both
10	3	From CLS	Names, quotes	Content words, punctuation	Moderately strong	Large	Second
10	4	From CLS	Names, verbs related to names	Content words	Strong	Large	Second
10	4	Between tokens (punctuation, function words and names to function words)	Auxiliary verbs, conjunctions, names, punctuation	Content words, function words, punctuation	Moderately strong	Medium	Both
10	5	From CLS	Full stops, pronouns, names	Content words, function words, punctuation	Moderately strong	Global	Both
10	5	Between tokens (names to pronouns)	Names, pronouns	Content words, function words	Moderately strong	Medium	Both
10	6	From CLS	Full stops	Punctuation	Moderately strong	Global	Both
10	7	From CLS	Verbs	Content words	Moderately strong, instance-specific	Global	Both
10	8	From CLS	Full stops, pronouns, adverbs, verbs	Content words, function words, punctuation	Moderately strong, instance-specific	Global	Both
10	9	From CLS	Names, punctuation	Content words, punctuation	Moderately strong, instance-specific	Large	Second
10	10	From CLS	Names	Content words	Moderately strong	Large	Second

10	11	From CLS	Names, verbs related to names	Content words	Moderately strong, instance-specific	Large	Second
10	12	From CLS	Full stops, commas, prepositions, names	Content words, function words, punctuation	Moderately strong	Global	Both
11	1	From CLS	Names, narrational verbs, stylistically variable words, diatopic variations, full/short form variations	Content words, function words	Moderately strong	Global	Both
11	2	From CLS	Names, adverbs, prepositions, pronouns, rare punctuation	Content words, function words, punctuation	Moderately strong	Global	Both
11	3	From CLS	Full stops	Punctuation	Moderately strong	Global	Both
11	4	From CLS	Full stops	Punctuation	Moderately strong	Global	Both
11	5	From CLS	Full stops	Punctuation	Moderately strong	Global	Both
11	6	From CLS	Full stops, pronouns, names	Punctuation	Moderately strong, instance-specific	Global	Both
11	7	From CLS	Full stops, pronouns, names	Punctuation	Moderately strong, instance-specific	Global	Both
11	8	From CLS	Names, punctuation, pronouns	Content words, function words, punctuation	Moderately strong, instance-specific	Global	Both
11	9	From CLS	Names, pronouns	Content words, function words	Moderately strong, instance-specific	Large	Second
11	10	From CLS	Names, narrational verbs, adverbs, pronouns, conjunctions, grammatical affixes, full stops, rare punctuation	Content words, subword tokens, function words, punctuation	Moderately strong, instance-specific	Global	Both, but mainly first
11	11	From CLS	Names, narrational verbs, adverbs, pronouns, conjunctions, grammatical affixes, full stops, rare punctuation	Content words, subword tokens, function words, punctuation	Strong, instance-specific	Global	Both, but mainly first
11	12	From CLS	Names, pronouns, adverbs, punctuation	Content words, function words, punctuation	Moderately strong, instance-specific	Global	Both, but mainly second
12	1	From CLS	Names, narrational verbs, punctuation	Content words, punctuation	Moderately strong, instance-specific	Global	Both, but mainly first
12	2	From CLS	Full stops, other punctuation, names	Content words, punctuation	Moderately strong, instance-specific	Global	Both
12	3	From CLS	Narrational verbs, function words, punctuation, uniform attention (certain regions)	Content words, function words, punctuation, uniform attention	Moderately strong, instance-specific	Large	Second
12	4	From CLS	Narrational verbs, function words, punctuation	Content words, function words, punctuation	Moderately strong, instance-specific	Large	First
12	5	From CLS	Narrational verbs, function words, punctuation	Content words, function words, punctuation	Weak	Global	Both
12	6	From CLS	Names, narrational verbs, punctuation	Content words, punctuation	Moderately strong	Global	Both
12	7	From CLS	Names, topical nouns	Content words	Strong	Large	First
12	8	From CLS	Names, topical nouns	Content words	Moderately strong	Global	Both
12	10	From CLS	Names, topical nouns, narrational verbs, stylistically variable words, diatopic variations, full/short form variations, infrequent words, various mistakes, punctuation	Content words, special function words, mistakes, punctuation	Moderately strong	Global	Both
12	11	From CLS	Names, topical nouns, narrational verbs, stylistically variable words, diatopic variations, full/short form variations, infrequent words, various mistakes, punctuation	Content words, special function words, mistakes, punctuation	Weak, instance-specific	Global	Both
12	12	From CLS	Full stops OR Names, topical nouns, narrational verbs, stylistically variable words, diatopic variations, full/short form variations	Content words, special function words, punctuation	Moderately strong, instance-specific	Global	Both