Explainability Techniques to Locate and Correct Grammatical Errors

Enrique Javier Cobo Jimenéz

Ronaldo Soares Rigobello

Department of Computer and Systems Sciences

Degree project 30 HE credits

Computer and Systems Sciences

Degree project at the master level

Spring term 2021

Supervisor: Ioannis Pavlopoulos

Stockholm University

Abstract

Since the advent of BERT (Bidirectional Encoder Representations from Transformers) in 2018, the field of Natural Language Processing has been transformed overnight. As a consequence, new inroads have been made in Grammatical Error Detection (GED) and Correction (GEC). Grammatical Error Detection is the task of identifying and locating grammatical errors in sentences, and Grammatical Error Correction is the task of transforming erroneous sentences into correct ones. The study aims to address the suitability of explainability mechanisms combined with a counterfactuals-based approach to solve GED and GEC tasks. Thus, the research question is *"How suitable are explainability techniques for solving Grammatical Error tasks?"*

To that end, we employed a quantitative research design, using experiments as the research method. A pipeline of three modules was designed to detect and locate errors and correct grammatically incorrect sentences. In Module 1, we fine-tune BERT on the task of sentence classification and employ its predictive capabilities in error detection. Next, in Module 2, we investigate the efficacy of LIME (Local Interpretable Model-Agnostic Explanations) as an explainability mechanism when applied to the task of locating errors in sentences. We also assess the efficacy of BERT's attention head scores in the same task. In Module 3, we take advantage of BERT's masked language modeling capabilities to generate candidate words to correct the grammatical errors. Finally, we use BERT fine-tuned to evaluate those candidates in terms of correctness to ensure they are actual counterfactuals.

Two classifiers were evaluated for grammatical error detection: BERT model fine-tuned on sentence classification and Random Forests (RF). With a 12-point accuracy lead over RF, BERT was chosen as the best classifier. Next, we evaluated LIME and attention head scores for error location; LIME proved being the best explainer with a 3-point accuracy lead over attention. Finally, a Masked Language Model (MLM) was used for the generation of candidate words to correct errors, which were evaluated against the fine-tuned BERT model. Overall, the system was able to produce contrasted correct versions of the input sentences detected as erroneous by Module 1 in at least 60% of the cases.

Keywords: GED, GEC, BERT, LIME, attention heads, Masked Language Modeling, Natural Language Processing, Machine Learning

Synopsis

Background	Natural Language Processing (NLP) algorithms enable computers and machines to process and analyze large amounts of natural (i.e. human) language data. With growing interest in this field in recent years, it is expected that NLP technology will play a much larger role in the coming years. In the field of education, there is a strong interest in technologies that support proficiency measures and content accuracy. Consequently, the NLP tasks such as Grammatical Error Detection and Grammatical Error Correction – whose aim is to identify and correct different types of grammatical errors in sentences – are becoming more popular.
Problem	There is a growing need for innovation that supports writing tasks requiring source use, argumentative discourse, and factual content accuracy. Such needs could be fulfilled by a GEC system capable of both correcting grammatical errors and providing feedback to users on what constituted or caused said errors.
Research Question	The aim of this study is to explore the usage and effectiveness of model explainability mechanisms in grammatical error correction tasks. Thus, the research question is formally defined as "*How suitable are explainability techniques for solving Grammatical Error tasks?*"
Method	This study employs experiments as the research method, yielding a software system artefact designed to address the research question. A three-stage pipeline architecture is proposed to 1) detect erroneous sentences, 2) locate the error in the sentence, and 3) suggest a correction.
Results	BERT outperformed RF in the sentence-level classification task, with AUC 84% vs 72% for RF. In the error word extraction task, LIME performed better than attention heads on the test set, with an accuracy score of 63% vs 60%, and a correlation analysis showed that LIME and attention heads work independently. In the Candidate Generation task, an MRR score of 52% was achieved on the test set. Overall, the system was able to produce contrasted correct versions of the input sentences detected as erroneous by Module 1 in 60% of the cases.
Discussion	This study contributes to the field of NLP by providing empirical findings of attention scores and LIME applied to error extraction in sentences allowing for GEC, which is a novel application in this field. A delimitation of the study was that only single-error sentences were evaluated, which greatly limited the size of the validation and test sets. Besides, lack of inter-annotator agreement might lead to having several candidates to correct the same sentence.

Acknowledgements

We would like to thank our supervisor, John Pavlopoulos, for his patience, dedication, and willingness to endure the many meetings throughout this thesis.

Similarly, we would like to acknowledge Katerina Korre for her willingness to spend hours annotating our results and sharing her knowledge with us.

Finally, we would like to thank our peers that took the time to review our work and give very interesting feedback that we then incorporated to the text.

Ronaldo: I would like to thank to my thesis partner, Enrique, for all the hard work, good humor, and collaboration.

I would also like to extend a big thanks and appreciation to my wife, for her patience, for always being there for me, for her loving support, and for the countless times when she was willing to pick up the slack watching our young child so that I could dedicate more time to the classes and this thesis.

In addition, I would like to thank my mother, for always believing in me and always being there for me.

Enrique: First off, my most enormous thanks to Ronaldo for his continuous effort and valuable support throughout the Thesis. We know it has not always been easy: our time-zone difference, countless meetings, overtime work, and many hiccups we found in our way. But, at least, we could always find ways to get around all this in a very positive way.

I would also like to thank my wife-to-be, Irene, for her endless patience and incessant, much-needed support. After all, it is not an easy task to be working and studying at the same time. Still, it was comforting to know that you were next to me, willing to give a helping hand.

Last but not least, I would like to thank my whole family, especially my parents, sister, and grandparents. Less spare time sometimes meant not being able to be there, so I just hope that, after you read the Thesis, you can see it was (at least a bit) worth it.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Acc	Accuracy
AccX	Top X Accuracy, where X is a non-zero positive integer
AI	Artificial Intelligence
ASR	Automatic Speech Recognition
AUC	Area Under the ROC Curve
BEA	Building Educational Applications
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional LSTM
CLS	Classification
CNIL	Commission Nationale de l'Informatique et des Libertés
CoNLL	Conference on Computational Natural Language Learning
DL	Deep Learning
ELMo	Embeddings from Language Models
ERRANT	Error Annotation Toolkit
EWE	Error-specific Word Embeddings
E&GWE	Error-specific and grammatical-specific Word Embeddings
FCE	First Certificate in English
GEC	Grammatical Error Correction
GED	Grammatical Error Detection
GWE	Grammatical-specific Word Embeddings
L2	Second language
LIME	Local Interpretable Model-Agnostic Explanations
LR	Logistic Regression
LSTM	Long Short-Term Memory
MLM	Masked Language Model
MRR	Mean Reciprocal Rank
NLP	Natural Language Processing
NMT	Neural Machine Translation
NSP	Next Sentence Prediction
OCR	Optical Character Recognition
OOV	Out-of-vocabulary
POS	Part of speech
RF	Random Forest
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic curve
SEP	Separator
SVM	Support Vector Machine
TF-IDF	Term Frequency Inverse Document Frequency

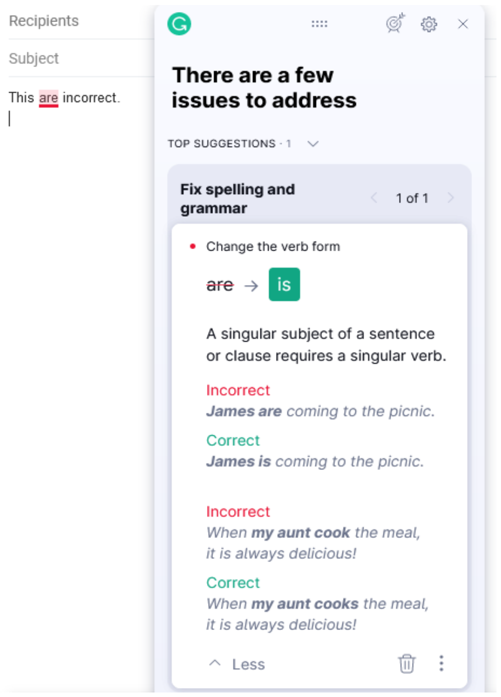
1 Introduction

1.1 Motivation

Natural Language Processing is essentially "focused on the design and analysis of computational algorithms" (Eisenstein, 2018, p. 2) necessary for the processing of natural languages (i.e. human languages). In other words, NLP applications seek to make human languages "understandable" to computers and machines and, thus, allow for new computational abilities, ranging from optical character recognition (OCR) to automatic grammatical error detection and correction.

A growing number of people are learning a second language (L2), with English and Chinese being popular first choices. Anyone who is bilingual or is learning a second language will attest to the fact that becoming proficient is not a trivial task and non-native (and sometimes even native) speakers often have trouble with particular aspects of foreign languages, such as the correct usage of prepositions in the English language. Essentially, the same can be said of any language. Hence, it is easy to see how useful an application capable of automatically detecting, flagging and explaining grammatical errors to learners would be. Such an application, or tool, would fall under the broad umbrella of NLP. Grammarly,[1] a commercial tool available online, is an example of such an application.

Grammarly is a proprietary online linguistic prescription tool based on Artificial Intelligence (AI) and NLP technologies (Grammarly, 2019). It is essentially a digital writing assistance tool, whose goal is to detect grammatical mistakes (as well as others, such as spelling, punctuation, etc.) in written English and to make suggestions to the users on how to correct them.



This are incorrect.

There are a few issues to address

TOP SUGGESTIONS · 1

Fix spelling and grammar

1 of 1

Change the verb form

~~are~~ → is

A singular subject of a sentence or clause requires a singular verb.

Incorrect
James are coming to the picnic.

Correct
James is coming to the picnic.

Incorrect
When my aunt cook the meal, it is always delicious!

Correct
When my aunt cooks the meal, it is always delicious!

Figure 1 Example of a Grammarly output

[1] *www.grammarly.com*

This tool is just an example of the applications and interest garnered by the NLP tasks of Grammatical Error Detection (GED) and Correction (GEC). In fact, there are new approaches under development such as error-type classification, i.e., the identification of the category to which the grammatical error belongs. Several error categories could be verb errors, preposition errors or concordance errors. However, the first step in error-type classification consists of detecting the error; a task which can still be improved.

In recent years, a powerful tool called BERT (Devlin, Chang, Lee, & Toutanova, 2018) has impacted the NLP field in general and, especially, grammatical error tasks. In a simplified manner, BERT is a transformer that has been heavily pre-trained, allowing the model to efficiently encode word and sentence contexts into the embeddings, which can then be used to solve many NLP-related problems, as it has been shown by Devlin et al. (2018).

1.2 Research problem

The literature defines three main NLP tasks: grammatical error detection, classification, and correction. These three tasks are closely intertwined, and it can be difficult to distinguish from among them. Ultimately, it could be argued that all three are simply steps within one single task, namely, grammatical error correction.

The problem that this work addresses is the development of a system for grammatical error correction tasks using state-of-the-art models and new techniques such as explainability. This thesis addresses a challenging problem, as there are many solutions yet no clear winner. It can also be considered of general interest since NLP tools and this system could be used for search autocorrect and autocomplete applications, proof-reading, virtual assistance like chatbots and speech recognition, among others. Finally, the fact that explainability techniques are employed to solve the grammatical error tasks is a novel approach.

This work first focuses on the problem of error detection and the two steps therein. For the binary-problem step, BERT looks as a suitable solution (i.e., a candidate sentence is fed to BERT as an input, and BERT returns whether the sentence contains a grammatical error or not). However, there are different alternatives to address where in the sentence the error is located, and one unexplored alternative is the use of model explainability techniques.

Lipton (2018) asserts that explainability is post-hoc interpretability. Miller equates explainability with interpretability and argues that it is the "degree to which an observer can understand the cause of a decision" (Miller, 2019, p. 8). Thus, model explainability is a criterion used to explain the reasoning behind a model's predictions and why a particular outcome was determined over another. Simply put, explainability affects user confidence in the predictions: being aware of the reasons why a particular prediction was made will, in turn, result in the user trusting or distrusting it.

Explanations can be intrinsic or post-hoc (Molnar, 2020). Models possessing simple structures, such as decision trees and logistic regression, are intrinsically interpretable and explainable (Molnar, 2020). As such, intrinsic explainability is also model-specific. Post-hoc, on the other hand, refers to explainability that is model-agnostic and is achieved after model training.

Secondly, this work also focuses on the problem of grammatical error correction. The approach we used to solve it is to use counterfactuals, which is another form of explainability (Molnar, 2020), to "suggest" alternatives to change an erroneous sentence into a correct one, while only using a minimal number of changes. “Minimal number of changes” in this context means the smallest number of erroneous tokens (i.e. words) that need to be changed to correct the sentence.

This work uses explainability mechanisms not only to affect user confidence. The artifact we propose is an exaptation, in such a way that the explanations will be used to locate the error within a sentence, as well as to serve as the basis to suggest possible corrections.

Finally, the development of new models for grammatical error tasks is full of challenges. Although the main tools and resources are publicly available and open source to encourage the development of the field, the real implementation in programming environments is not straightforward. Besides, the choice of the corpora impacts the performance as well, both in the learning phase of the models and in the performance evaluation.

1.3 Research Question and goals

The research question of this thesis is defined as follows:

How suitable are explainability techniques for solving Grammatical Error tasks?

In order to answer it, the Thesis consists of the following goals:

- Develop and train a model capable of detecting if a sentence contains an error. This task is also referred to as sentence-level classification. This is a binary classification problem: the sentence as a whole can either be correct or incorrect.
- Test the model to verify that it can accomplish the sentence-level classification task.
- Employ model explainability mechanisms to locate the error. This task is also referred to as word extraction. This way, the explanations are used as a learning sub-product since the model was never trained to pinpoint which tokens cause sentences to be incorrect.
- Compare different explainers to determine which is the most suitable for the word extraction task.
- Develop a counterfactual-based Module to suggest sentences classified as correct by the sentence-level classifier developed earlier in this work.

1.4 Delimitation of the study

The current study will be limited to corpora in written (as opposed to spoken) English, due to their wide availability.

Another delimitation of this study is the use of erroneous sentences containing a single grammatical error that can be corrected by replacing a single word. These constitute the simplest errors to identify and correct because we impose that 1) there is always one and only one erroneous word in the sentence that can only be corrected by 2) proposing one-word suggestions.

Furthermore, the work and its underlying models will be case insensitive, meaning that the sentences will always be converted to lower case before being fed into the models. This does not represent a loss of generalization since case-related errors can, in most cases, be detected by a rule-based classifier with the capitalization requirements for the language. Such errors include, for example, the use of the first-person pronoun in lowercase (i.e. “i” instead of “I”) and sentences beginning with uncapitalized letters. Both cases can be solved via simple rules. Therefore, by constraining the sentences to be lowercased, we ensure that the model learns more complicated language features.

A final delimitation is the use of open-source tools whose code is readily available to make the results reproducible.

1.5 Thesis structure

The contents of this paper are structured as follows:

- Chapter 2 presents the extended background as well as some related work useful to understand the Thesis and to provide some context to what has been done in the field prior to this work.
- Chapter 3 introduces the methodology, which represents the research strategy and the motivation behind the experiments that we carried out. Besides, some practicalities around how the work is conducted are also presented.
- Chapter 4 includes details on the experiments that we carried out together with the results that we obtained. A discussion of the results will also be presented.
- Chapter 5 summarizes the work that has been done and presents some future research lines that arose as a consequence of the Thesis. Conclusions and limitations will also be presented.

2 Background

2.1 Grammatical Error Tasks in NLP

Grammatical Error Detection, Correction and Classification are three tasks belonging to NLP that were outlined in Chapter 1.

2.1.1 Grammatical Error Detection

The first step in the grammatical error correction pipeline is to establish whether a given sentence in a text contains a mistake or error and to locate it within the sentence. The task is made more complex whenever multiple errors are present in the same sentence, which is often the case. Grammatical Error Detection is thus the task of analyzing sentences of a text to determine whether they contain grammatical mistakes and to pinpoint their locations. This can be done on sentence- or word- (i.e. token) levels.

GED at the sentence level is about establishing whether the sentence as a whole is erroneous and is essentially a binary task: the output is usually positive (i.e. the sentence contains one or more errors) or negative (i.e. the sentence is correct). Conversely, GED at the token level is where individual words (i.e. tokens) are used as input to determine whether they are causing an error in the sentence, in which case their starting and ending offset positions – hence, the location of the error – can be determined.

Common machine learning approaches employed in GED systems in the past included baseline classifiers such as Support Vector Machines (SVMs), Decision Trees, and Logistic Regression; however, more recent ones employ more advanced methods, such as neural networks.

2.1.2 Grammatical Error Correction

Put simply, grammatical error correction is the task of transforming an erroneous sentence into a correct one, all the while ensuring that the original meaning of the sentence is preserved. However, this is not always a straightforward task (Chodorow, Dickinson, Israel, & Tetreault, 2012). Very often, a sentence contains multiple errors, and in different locations.

A further complication is the fact that often, multiple solutions are possible. As an example, take the following erroneous sentence “This are incorrect”. This could be corrected in two ways: (1) “This is incorrect”; (2) “These are incorrect”. Without knowing the intended meaning (i.e. was the author referring to one single sentence or several?), it is impossible to determine whether one solution is more suitable than the other. Note, however, that a) both solutions comprise the same number of changes (i.e. both require a single change) to transform the erroneous sentence into a correct version; and b) where the first correction changes the verb conjugation (i.e. “are” → “is”), the second changes the demonstrative pronoun (i.e. “this” → “these”) instead. This is a simple example; however it illustrates well some of the challenges in GEC.

Other issues are redundant or missing words, which greatly increase the difficulty of this task. Furthermore, there are cases when errors are not actually errors, which can happen due to e.g. the existence of different style manuals at newspapers (Napoles, Sakaguchi, Post, & Tetreault, 2015).

Approaches to GEC tasks typically fall into one of three basic categories: rule-based, classification-based, and machine translation (Ailani, Dalvi, & Siddavatam, 2019). Rule-based systems rely on POS (part of speech) taggers, grammar rules and parsers to output error types. Classification-based systems are normally composed of machine learning classification algorithms that are trained on error-coded corpora. Both rule and classification-based systems are either unable to deal - or perform poorly - with more complex or multiple error types occurring in the same sentence (Ailani et al., 2019). Machine translation approaches can be further subdivided into statistical and neural approaches and can be used to solve any error types.

2.1.3 Grammatical Error Classification

Once it has been determined that errors are present in a sentence and a correction is provided, those errors can be classified. Several classification frameworks have been developed over the years (e.g. Nicholls, 2003; Dahlmeier, Ng, & Wu, 2013) with different - and largely arbitrary - designs. In the context of this work, error types are determined by the grammatical function of the word(s) involved via POS analysis and that errors are tagged by type (Granger, 2003). These tags are typically produced by annotators via a standardized system of error tags.

Such is the case with ERRANT (ERRor ANnotation Toolkit). ERRANT is a dataset-agnostic tool which classifies annotators' edits according to a rule-based error type framework (Bryant, Felice, & Briscoe, 2017). It works by aligning the parallel original and corrected versions of sentences, allowing for the edits/annotations to be extracted and tagged according to the rules' framework, along with offsets for the edits, among other information. The underlying framework relies on 50 rules and breaks down errors into 25 main types, which can be further categorized into three subtypes, depending on whether the annotator's edit is a replacement (e.g. R:), missing (M:) or unnecessary (U:) tokens. An overview of these main error types, along with their descriptions, is provided in Appendix A.

2.2 Recent trends on GED and GEC: From BiLSTM to BERT

The rapid development of GED makes systematic reviews of approaches and trends a cumbersome matter since the number of works published every year can render any review quickly outdated. Some attempts to gather grammatical error checking systems have already been made (Grundkiewicz, Bryant, & Felice, 2020; Madi & Al-Khalifa, 2018). Several of such systems rely on earlier techniques, such as statistics or rules for syntax and morphology. Those approaches are mainly used now in GEC systems for complex languages other than English, such as Latvian (Deksne, 2016), Punjabi (Kaur, Garg, & Tech, 2014), or Bangla (Hasan, Hozaifa, & Dutta, 2014). The corpus available for these languages, i.e., the language resources consisting of structured sets of texts, is more limited. On the other hand, deep neural networks have mostly been applied to the written English corpora, although some works extend them also to other languages, such as Arabic (Alkhatib, Monem, & Shaalan, 2020).

The main advantage of using deep learning models compared with traditional machine learning is that these neural networks can learn automatically and even make decisions, overcoming time-consuming activities such as statistical training with parts-of-speech or parsing (Madi & Al-Khalifa, 2018). When it comes to deep learning applied to GEC and GED, two different research trends have gained momentum in the last five years: neural machine translation and sequence labeling for GED.

Neural machine translation (NMT) translates sentences from one language to another, using neural networks, particularly Recurrent Neural Networks (RNN). This line of research was established and represented by (Cho et al., 2014) and (Bahdanau, Cho, & Bengio, 2016), who used RNN encoder-decoders to translate from English to French.

2.2.1 BiLSTM architecture

In 2016, Rei and Yannakoudakis published a groundbreaking work for error detection in written texts. Compared to the previous works which were error-type specific, Rei and Yannakoudakis developed a general error detection system using neural networks. They compared the performance of several neural network architectures applied to the same set of data to conclude that the BiLSTM was able to achieve the best results.

LSTM (short for Long Short-Term Memory) (Hochreiter & Schmidhuber, 1997) is a special subtype of RNN, capable of learning long-term dependencies. These networks were explicitly designed to overcome the problem of long-term dependency of RNN, i.e., the need to use more context to learn and connect the information (Alkhatib et al., 2020). Refined LSTMs had already been applied to tasks such as speech recognition (Graves, Mohamed, & Hinton, 2013) or machine translation (Luong, Pham, & Manning, 2015); however Rei and Yannakoudakis improved upon LSTMs to build a Bidirectional LSTM (or BiLSTM). The underlying idea is that one LSTM keeps the context of previous words, whereas the other *adaptatively* decides which information to include when performing the error detection.

Following the publication of the work in (Rei & Yannakoudakis, 2016), the BiLSTM approach has been extensively used for GED. It has been extended to other languages rather than English (Alkhatib et al., 2020), refined with character-level information (Rei, Crichton, & Pyysalo, 2016), enhanced with pre-training models (Wang & Tan, 2020), or even applied to spoken language (Knill, Gales, Manakul, & Caines, 2019), among others.

These works use several corpora to test error detection capabilities. The most important ones are FCE and CoNLL13 or 14, both in written English. FCE comes from exam scripts of English learners taking the First Certificate in English (Yannakoudakis, Briscoe, & Medlock, 2011). This is also the corpus used on this work, and is further described in Chapter 3. CoNLL13 (Ng, Wu, Wu, Hadiwinoto, & Tetreault, 2013) and 14 (Ng et al., 2014) are datasets written by higher proficiency English learners on different technical topics at the Conferences on Natural Language Learning in years 2013 and 2014 respectively.

2.2.2 Pretraining with contextual embeddings: BERT

The introduction of BERT[2] (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) caused a revolution in the field of grammatical error correction tasks and NLP in general. They introduced contextual embeddings: word representations that are built based on the context in which the word appears (Bell, Yannakoudakis, & Rei, 2019) .

BERT is designed to be used for bi-directional pre-training – in contrast to the classical left-to-right pre-trainings. Consequently, it can be applied in many NLP tasks and not only in GED. It advanced the state of the art of pre-training, surpassing the performance of predecessors such as ELMo (Peters, Ammar, Bhagavatula, & Power, 2017) or Flair (Akbik, Blythe, & Vollgraf, 2018), thus becoming the de-facto state-of-the-art solution.

[2] *https://github.com/google-research/bert*

In fact, the basic BiLSTM model for GED has been texted with embeddings from BERT, ELMo, and Flair (Bell et al., 2019). The former yields the highest performance regardless of the dataset (FCE or CoNLL, among others). One possible explanation is the different training corpora behind each contextual embedding model. ELMo and Flair embeddings are trained with One Billion Word Benchmark (0.8 billion tokens of training data), whereas BERT is trained with the BooksCorpus and English Wikipedia (3.3 billion tokens of training data).

Due to the impact of BERT in NLP tasks, efforts have been made to try to better understand the reasons behind its good performance. Such works include Kovaleva, Romanov, Rogers, & Rumshisky (2019), which offered a detailed analysis of BERT, remarking that attention is key on BERT, proposing a method to improve it. Similarly, Clark, Khandelwal, Levy, & Manning (2019) also describes which head and layer combinations are good at finding, for example, grammatical categories (such as prepositions) or grammatical functions (such as coreferences or direct objects). Nevertheless, a recent work by Kobayashi et al. (2020) challenges it and suggests that BERT lacks enough attention in some cases, and that even attention by itself is not enough. More factors are required to guarantee a valid output.

Since BERT is the current state-of-the-art tool for NLP tasks, it is the tool used in this study. Its internals are presented in the following section.

2.3 BERT: Bidirectional Encoder representations from Transformers

In Section 2.2, we presented some related work that has been done in the context of BERT. In addition, and because its introduction represented a groundbreaking achievement, many educational sources have also been written on the topic. Examples of popular blog posts explaining what BERT does and how it works are (Alammar, 2018), (Futrzynski, 2020), and (McCormick, 2019).

Attention is a mechanism used by neural networks to mimic cognitive attention from the human brain. In short, it serves to amplify important parts of the input data, while down-playing unimportant parts. In essence, how BERT works can be simplified to the transformations done by the attention heads. There are many heads per layer (or transformer), and BERT consists of many such transformers. The number of each of these is determined by BERT’s size, which we discuss later.

It starts with the Input embeddings, which is the representation of the incoming sentence in a format that BERT understands. For that, there is a mandatory preliminary step: tokenization, which splits the input sentence into elements (so-called *tokens*).

Note that several special tokens need to be defined: [CLS], which stands for Classification, is added at the beginning of each incoming sentence and is intended to serve as an input to a classifier. [SEP], which stands for Separator, is introduced after every sentence. [MASK] is used whenever a placeholder to a single token needs to be introduced, which is the case for the masked Language Model (MLM) task explained later in this chapter.

For each token, the input embedding aggregates the following information: the token embeddings, segment embeddings, and position embeddings. Token embeddings are obtained from WordPiece (Wu et al., 2016), with a vocabulary size of 30,552 tokens. It also breaks down composed words into different tokens (for example, “playing” becomes “play ##ing”), where ## denotes that the current token is linked to the previous one. This allows previously unseen or incorrect tokens (such as spelling mistakes) to be inserted.

Segment embeddings are only used whenever the [SEP] token is used. It represents the part of the phrase to which the token belongs. Every time that [SEP] is found, the segment embedding is incremented. Position embeddings encode information regarding where to find the corresponding token in the sentence. In other words, it keeps a counter of the tokens in the sentence.

Figure 2 shows the final input embeddings for the example sentence "My dog is cute. He likes playing". For each token, the input embedding consists of a number of features that depend on BERT's size.

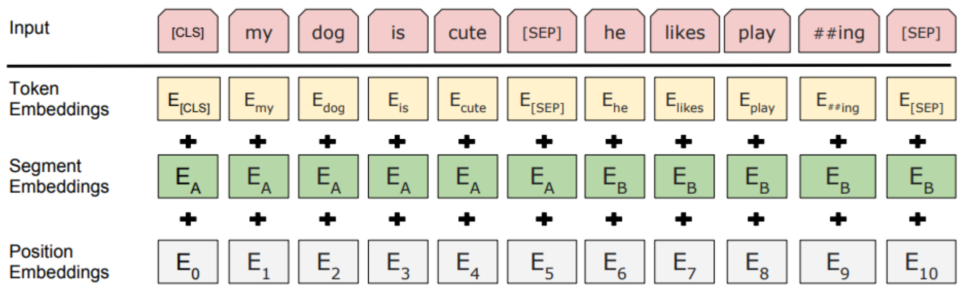

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8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

Figure 2 Embeddings for the sentence "My dog is cute. He likes playing." (Devlin et al., 2018)

Once the input embeddings are ready, the processing can begin. The next step is to create linear projections of the input embeddings into three new vectors: keys, queries, and values. The aim behind these linear projections is a) to reduce the input space to something more manageable (these new vectors contain 64 features) and b) to map the selected component on which we want the attention head to focus, since this will vary from head to head. For instance, one key value could represent whether the projected input embedding is a preposition, and a query could represent where the key could be located in the input embedding.

The newly created key and query vectors are combined via their scalar product. A nonlinear function denoted Softmax is then applied to the resulting matrix, which highlights which component is the most important, assigning to it a higher weight. The result of this operation is referred to as the attention weights.

Finally, the last projection, value, comes into play. It could represent, for example, whether the original input embedding relates to physical places. Then, the result from Softmax is linearly combined with the values vector to create the contextualized embedding, representing the output of the attention head. The whole process is exemplified in Figure 3.

This is, in summary, what each of BERT's attention heads does. The attention heads are replicated many times per transformer; and BERT instantiates many of these. The contextualized embeddings resulting from one attention head are concatenated to become the input embeddings for the next transformer.

BERT is released in many different sizes, each of which varies the number of attention heads, transformers, and hidden features. This work uses BERT Base uncased, which consists of 12 attention heads per transformer, 12 transformers, and 768 features per token. Uncased refers to the fact that the tokenizer will, as a first step, ensure that all the incoming words are in lower case.

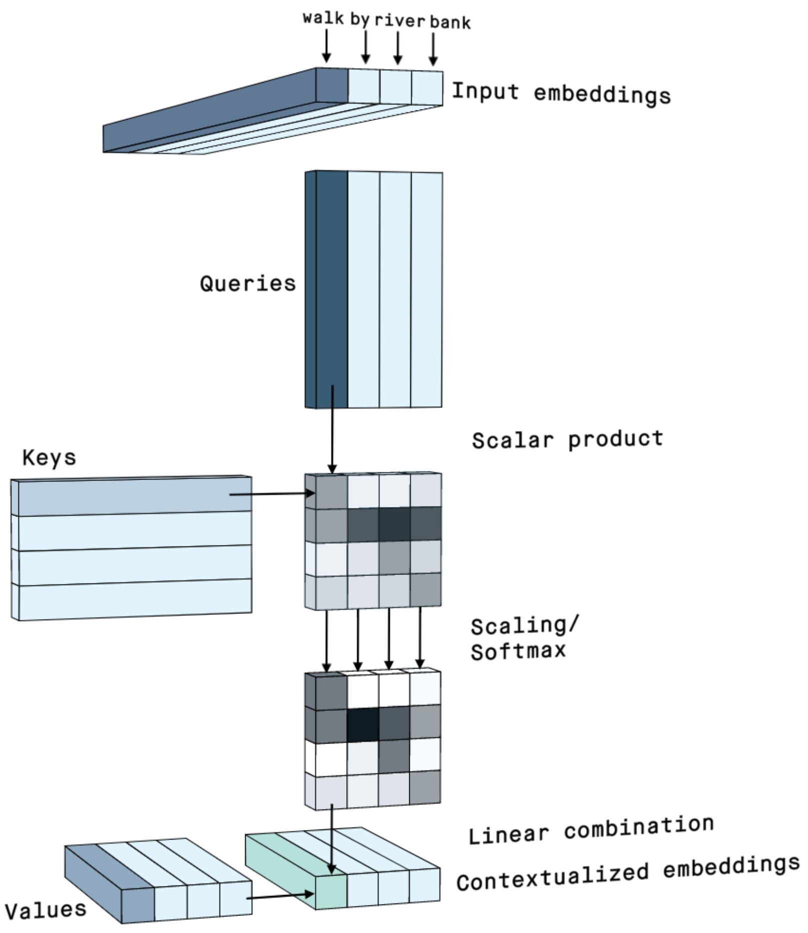


Figure 3 Representation of the operations performed by an attention head. (Futrzynski, 2020)

Figure 4 shows what BERT looks like from a high-level perspective. It has been adapted to reflect the case for BERT Base. It inputs an arbitrary number of input embeddings, in this example 512, each of which consists of 768 features (represented by the blue array). Each input embedding traverses the different transformers (12 for BERT Base), getting context as it goes by. Finally, BERT outputs contextualized embeddings, as many as there are input tokens.

BERT is instantiated in a pre-trained state, so that most of the work to teach the different relations between words in a language has already been done. The reason for that is that the pre-training process is memory and time intensive; however, once it is done for a language, the pre-trained model can be exported and reused. In fact, BERT was pre-trained on Wikipedia, which contains a large amount of tokens and sentences that have been fed into the network for a large number of epochs, of the order of millions of times.

Pre-training is achieved by means of two tasks: masked Language Model (MLM) and Next Sentence Prediction (NSP). The MLM procedure consists of masking away a certain number of tokens (15% in the case of BERT) randomly, which BERT is then supposed to predict. On the other hand, NSP refers to the task in which two sentences are fed into the systems: half of the time the second sentence will be the continuation of the first one, while for the other half it will just be another sentence. BERT is trained to be able to classify whether one of the sentences logically follows the other.

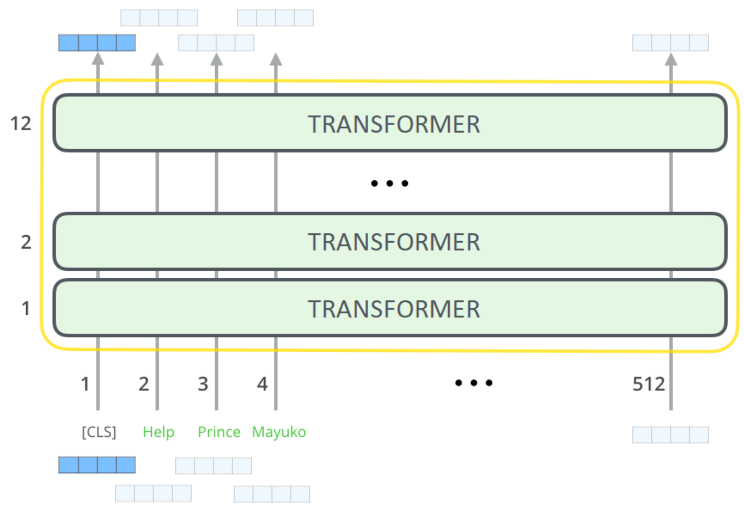
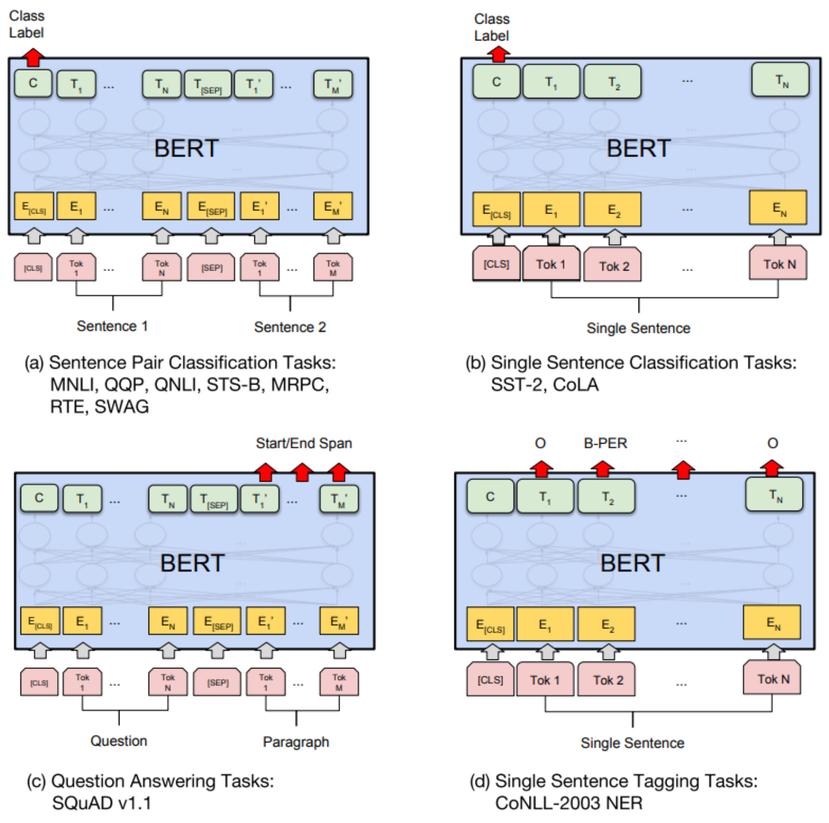


Figure 4 BERT, end to end. Modified from (Alammar, 2018)

Once pre-trained, BERT can be used to solve a wide variety of NLP tasks, as presented in Figure 5. There are four main use cases: use case a), or sentence pair classification task, where the idea is to differentiate whether Sentence 2 is a continuation of Sentence 1. In this case, the special token [SEP] comes into play. The output is obtained from the [CLS] token.

Use case b), or single sentence classification task, allows us to determine whether the sentence verifies one property or not. Examples of properties could be sentiment classifiers (e.g., to identify if a sentence contains a good or a bad review) or sentence level classification (to determine whether the input sentence contains one (or more) grammatical error or not. Again, the output corresponds to the contextualized embeddings of the [CLS] token.



Class Label
BERT
Sentence 1
Sentence 2
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG
Class Label
BERT
Single Sentence
(b) Single Sentence Classification Tasks: SST-2, CoLA
Start/End Span
BERT
Question
Paragraph
(c) Question Answering Tasks: SQuAD v1.1
O
B-PER
O
BERT
Single Sentence
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 5 BERT use cases as summarized in Devlin et al. (2018).

The question-answering task, presented in use case c), pretends to answer the question posed in the Question segment, with the context provided in the Paragraph segment. In this case, the output is obtained from the embeddings going from the [SEP] token onwards.

Finally, another example is single sentence tagging tasks, as explained in use case d). Here, the idea would be to classify or tag each incoming token.

All these different use cases can be achieved through the so-called fine-tuning process. Unlike pre-training (which requires a more substantial dataset), this step can be performed with a smaller dataset representing the task to be accomplished - for example, sentiment classification in movie reviews as in use case b) - so that it can learn how to produce the correct output. BERT is typically fine-tuned in less than one hour since the heavy work was already done during the pre-train phase. Since we are providing BERT with labeled examples, the fine-tuning process belongs to the supervised training domain.

2.4 Baseline classifier: Random Forests

This section presents Random Forests, a general-purpose model that has been used as a baseline, against which we later compare the results of the dedicated model.

Ensembles are combinations of machine learning models (i.e. classifiers), in which the individual decisions of those classifiers are combined in some way, resulting in more powerful models (Dietterich, 2000). In this way, Random Forests (RFs) are ensembles of decision trees in which a result (i.e. prediction) is reached by "majority rule", in classification tasks; or by averaging, in regression. Majority rule here means that each individual tree in the ensemble votes for a class and the class with the most votes determines the classification for the ensemble as a whole.

Individual decision trees are generally good at classification tasks; however they are prone to overfitting on only a part of the data. RFs turn this weakness into a strength by combining the predictions of the individual trees in the ensemble and thus greatly reducing the chances of overfitting (Breiman, 2001). Each classifier (i.e. decision tree) in the ensemble is generated randomly to ensure its uniqueness in the ensemble. This is ensured by two different techniques employed by Random Forests when generating trees: bootstrapping/bagging (Breiman, 1996) and random subspace method (Ho, 1998).

Bootstrapping is a method to ensure that each decision tree is built on a slightly different dataset (i.e. a bootstrap sample). It works by randomly drawing data points with replacement, where the same data point can be picked multiple times. The resulting bootstrap sample is of the same size as the original dataset, however around one third of the original data points are missing.

The random subspace method ensures that each decision tree is split (and thus, built) differently. When generating a tree, at each node, only a random subset of features is made available. Therefore, instead of testing from among all features for the best split, each node has to decide for the best split from among the random subspace made available to it.

Together, these two techniques - namely, random subspace and bagging - ensure that each individual decision tree is unique.

RFs are one of the most popular machine learning methods for classification tasks. Furthermore, they are simple yet very powerful and typically work very well without heavy tuning of parameters. For all these reasons, they were very attractive as a baseline classifier for use in this thesis work.

2.5 LIME: Local Interpretable Model-Agnostic Explanations

Single evaluation metrics often only provide an incomplete description of real-world tasks, as shown in the works of Ribeiro et al. (2016) and Doshi-Velez and Kim (2017). Compounding this problem, there is the fact that many machine learning models are seen as functional black boxes, whose rationale for reaching certain predictions or outcomes is nearly impossible to be grasped by users. Moreover, more abstract features such as nondiscrimination cannot be adequately quantified and assessed (Doshi-Velez and Kim, 2017). All of these can result in users not trusting the model itself or the predictions it makes which, in turn, can lead to users not using applications employing such models (Ribeiro et al., 2016). This is even more so in high-stakes applications, such as healthcare or bank loans.

Released in 2016, LIME (short for "Local Interpretable Model-Agnostic Explanations") is a model-agnostic post hoc algorithm that can be used to explain predictions locally in a faithful way (i.e. interpretable and understandable to humans) (Doshi-Velez and Kim, 2017). Being model-agnostic, LIME is not able to "peek" into the underlying model it is applied upon. Instead, LIME attempts to learn a local model that should be a good local approximation of the underlying machine learning model it is applied upon. This learned model, however, does not necessarily need to be a good global approximation (Molnar, 2020).

By relying on the basic assumption that even more complex models are still linear on a local scale, LIME learns a linear regression model that is a good approximation of the model in the neighborhood around the input to be explained (e.g. in the context of this thesis, the input is the original sentence or text being classified). This means that the explanations produced by LIME are locally faithful around the input (as opposed to globally) (Ribeiro, 2016).

LIME achieves this by making use of permuted samples (also known as local surrogate models) in the neighborhood of an input and tracking how the prediction of the original classifier changes. The neighborhood of a prediction is defined via an exponential smoothing kernel function that measures the proximity (Ribeiro et al. recommend the use of the cosine distance for text data) between the permuted sample and the original input (Molnar, 2020). The size of the neighborhood itself is defined by the kernel width: the larger the width the larger the impact farther permutations will have on the linear model.

For text data, permuted samples are generated by randomly masking (or removing, depending on how LIME is configured) one or more words/tokens from the original text. Essentially, the smoothing kernel function measures how close (i.e. similar) the permuted samples are to the original text: the more words masked from the sample, the farther the two sentences are from each other. The permuted samples are weighed as to their proximity to the original text and, along with their predictions from the original classifier, are then used to learn an explainable weighed linear model that approximates the original model in the vicinity of the input (Ribeiro, 2016).

An example of a LIME output in the context of an NLP application is provided with the sentence "A security guard saw it and put his attention on him". As a first step, LIME produces a configurable number N of permuted samples based on the original sentence. This is shown in Table 1.

Table 1 – Example of a LIME permuted sentence. Masked words are highlighted in black

Sample no.	Permuted sentence
1	A ■ guard saw it and put his attention on him
2	■ security guard saw it ■ put his ■ on him
…	…
N-1	A ■ ■ ■ it and ■ his ■ on ■
N	■ security guard saw ■ and ■ his attention ■ him

Each of these permuted sentences is then evaluated by the underlying model, and this information is used to construct the LIME's internal linear regression model. After this analysis is done, LIME outputs the results shown in Figure 6. The first thing to notice is that the underlying model defines the original sentence as erroneous with a probability of 81%. Next, it presents the explanation on why the sentence was classified to be incorrect. In this case, it highlights "on" as the main reason why the original sentence was classified as incorrect. Note, though, that it does not imply that "on" is the only word in the original sentence causing the sentence to be incorrect.

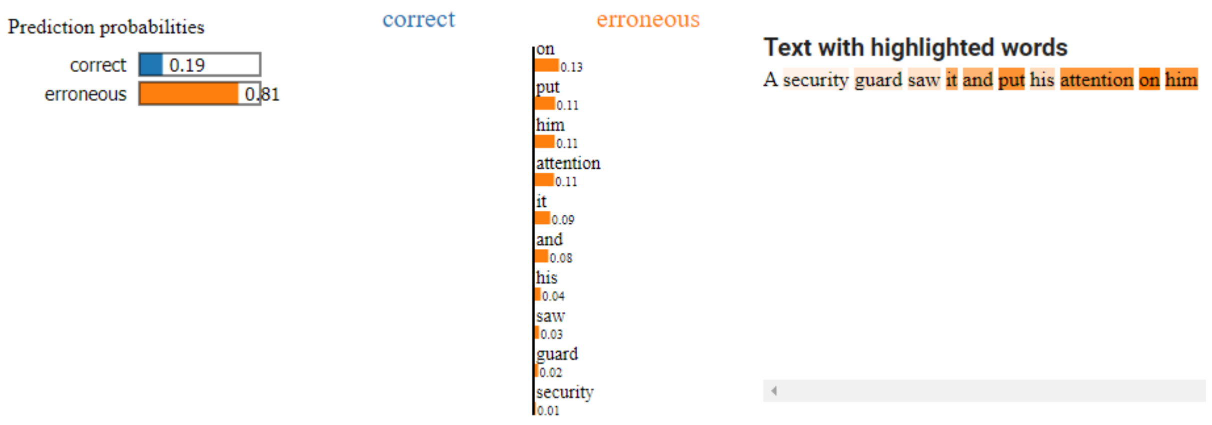


Figure 6 Example of a LIME output

This example illustrates how LIME can be useful, as it can provide further evidence that predictions produced by the model actually make sense.

Regarding the drawbacks of LIME, the typical criticism is directed at (1) its instability, where similar points may have very different explanations (Alvarez-Melis & Jaakkola, 2018); and (2) low fidelity of explanations (Rahnama & Boström, 2019).

2.6 Counterfactuals

In the field of machine learning, counterfactuals can be employed as a model-agnostic method to produce explanations for models (Carrillo, Cantú, & Noriega, 2021; Wachter, Mittelstadt, & Russell, 2017). As such, they are key components in systems being used to inform "societally critical domains, such as finance, healthcare, education and criminal justice".[3]

Simply defined, a counterfactual denotes a condition that would have been true under different circumstances and is often denoted as "If X had not occurred, Y would not have occurred" (Molnar,

[3] *https://github.com/interpretml/DiCE*

2020). In the context of our work, "For a given grammatically incorrect sentence, if token X was used instead of token Y, the sentence would be **correct**" (Molnar, 2020). A counterfactual can also be described as the "**smallest change to the feature values that changes the prediction to a predefined output**"[4].

In the context of this thesis, a counterfactual would be the smallest change to a grammatically incorrect sentence that would result in it no longer being predicted as erroneous. Therefore, in the sentence "*These* is an example", a counterfactual would be "This is an example", where "These" is changed to "This" for subject-verb agreement. For contrast, the counterfactual "These are examples" would require changing "is" → "are", "example" → "examples" as well as removing "an", thus failing to fulfill the smallest-change requirement.

2.7 Metrics used in evaluation

The following metrics are used to evaluate different aspects of the system. They are briefly introduced in this section to ensure that the present work is self-contained. The reader may decide to skip this part if already familiar with those.

2.7.1 Confusion Matrix

The confusion matrix helps us to understand the performance of a classifier after a test dataset has been applied to it. It consists of a square matrix of order N, where N denotes the number of classes in the dataset. For the case of a binary classifier, in which there are only two classes (typically denoted "positive" and "negative"), the confusion matrix looks as depicted in Table 2:

Table 2 – Example of a confusion matrix for a binary classifier

		Predicted	
		Negative	Positive
Actual	Negative	True Negatives (TN)	False Positives (FP)
	Positive	False Negatives (FN)	True Positives (TP)

Each column represents whether the classifier predicted certain sample to be negative or positive, respectively. For example, a sample in the first column indicates that the sample was classified to be "negative". Similarly, each row represents the true class of the sample. So, if a sample appears in the second row, it indicates that the actual class of the sample is "positive".

When combining rows and columns, four combinations of predicted vs actual are defined:

- A True Negative occurs when the sample is negative, and it was predicted to be "negative".
- A False Positive is obtained if the sample is negative, yet it was predicted to be "positive".
- A False Negative occurs when the sample is positive, yet it was predicted to be "negative.
- A True Positive is obtained if the sample is positive and it was predicted to be "positive".

[4] *https://christophm.github.io/interpretable-ml-book/counterfactual.html#example-8*

These four values can now be used to derive other metrics. For this work, the most important ones are accuracy, true positive rate, and false positive rate. The Accuracy of a binary classifier is defined as the total number of samples classified correctly (regardless of whether they belong to either of the classes) divided by the total number of samples. It ranges between 0 and 1, and a high value represents that most samples were classified correctly.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

The True positive rate is the ratio between the true positives and all sentences that belong to the positive class. This can also be referred to as recall of the positive class (or simply, recall):

$$TP\ Rate = \frac{TP}{TP + FN}$$

Similarly, the false positive rate is the ratio between the false negatives and all sentences that belong to the negative class.

$$FP\ Rate = \frac{FP}{FP + TN}$$

2.7.2 AUC, Area under the ROC curve

We start by defining ROC to be the Receiver Operating Characteristic curve, which is a graph that plots the True positive rate vs the False positive rate for different classification thresholds. The importance of the classification threshold comes from the fact that, the lower the threshold, the more items will be classified to be positive, hence increasing both the True positive rate and the False positive rate.

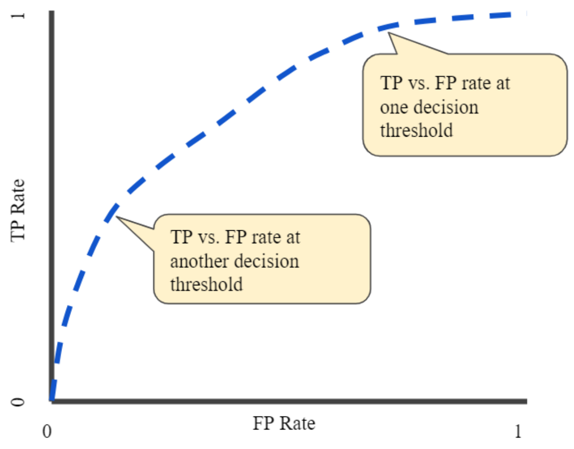


Figure 7 Example of a ROC curve[5]

The AUC, or Area Under the ROC curve, is the area captured between the X axis and the ROC curve above it. It determines the probability that the classifier classifies a random positive sample more highly than a random negative example. It ranges from 0 to 1, 0 being a model whose output is 100% wrong, and 1 being a model whose output is 100% right. A value of 0.5 is identified with a model whose output is completely random, i.e. it outputs positive or negative with 50% change.

[5] *https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc*

The AUC and Accuracy serve similar purposes; they determine how well a classifier performed. Nevertheless, the AUC performs best if the dataset is imbalanced, meaning that the actual number of samples from either class is significantly higher than the samples from the other class.

2.7.3 Top N Accuracy

In a multi-label classifier, the Top N Accuracy measures how many samples were classified correctly if the correct class is ranked in **at most** the n-th position. This represents a generalization of the accuracy for which errors in the first position can still be considered acceptable if they are found in the topmost positions of the classifier. Note that, indeed, Top 1 Accuracy is the same as Accuracy.

As an example, we consider that each word in a sentence is a different class. Suppose that we pass the sentence “This is how top N accuracy works”, and the classifier returns the classes in the following order: “N”, “works”, “This”, “accuracy”, “how”, “is”, “top”. If the class that we were expecting to get was “works”, top 1, 2, and 3-accuracy would be 0%, 100%, and 100%, respectively.

2.7.4 MRR, Mean reciprocal rank

In short, Mean Reciprocal Rank (MRR) is a statistic measure to evaluate systems that return a ranked (i.e. ordered by the probability of correctness) list of answers to queries (Q). It is calculated by the following formula, where rank_i is the reciprocal rank of the first correct answer for the *i*-th query. In cases where none of the answers are correct, the reciprocal rank is zero.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

For clarity, consider the following example in the context of a GEC application. In each row we have a grammatically incorrect sentence where we are looking for the correction (i.e. the query), a correct answer, the answers to the query and the reciprocal ranks. The resulting MRR for this example is: (1/1 + 1/2 + 1/3) / 3 = 0.61.

Table 3 – MRR example

Query	Correct answer	Query answers	Reciprocal rank
“I have *know* her for five years.”	known	<u>known</u>, loved, had	1/1 = 1
“I've always loved music so working *in* a concert was great.”	at	on, <u>at</u>, out	1/2 = 0.5
“I learnt from *you* letter that I've won the first prize.”	your	the, this, <u>your</u>	1/3 = 0.333

3 Methodology

3.1 Research strategy

A strategy can be defined as the “plan of action designed to achieve a specific goal” (Denscombe, 2014, p. 3). In research, working on the strategy means describing the methodological level, i.e. the high-level framework that provides better grounds to address the research question (O’Leary, 2017).

Since we are working in the field of computer science and engineering, the research strategy belongs to the pure quantitative approach, where the research highly relies on quantified data (O’Leary, 2017). In this study, we are interested in investigating causal and effect relationships when tuning the individual Modules designed to answer the research question. Therefore, this study employs **experiments** as the chosen research strategy, as described in (Johannesson & Perjons, 2014).

In this work, we have also taken the build approach described in Elio et al. (2011), which consists of building an artifact – a software system in our case – to answer the research question. As Elio et al. (2011) proposes, there are a series of good practices when choosing the Build methodology:

- Design the system considering a modular approach. In our case, we have started from a schematic diagram to define each of the sub-problems to be addressed, as can be seen in Figure 8. Our approach is to consider that error correction precedes error classification, although the classification can also be used as an input to the correction. This alternative path is depicted in dashed lines. Our study focuses on developing and evaluating a system based on different modules to accomplish the first two tasks: error detection and correction. Our approach was to reach a certain degree of modularity, focusing first on the sentence identification, then on the word extraction, to end with a corrected sentence. Also, with the Module structure, we ensure that each function can be executed separately, and that different building blocks can be exchanged to construct a different pipeline.

- Reuse components. As presented in previous chapters, one of the main parts of the work is the use of BERT. As we have seen in chapter 2, since its release in 2018, it has been a necessary tool for most NLP tasks, and it is now part of all state-of-the-art solutions.

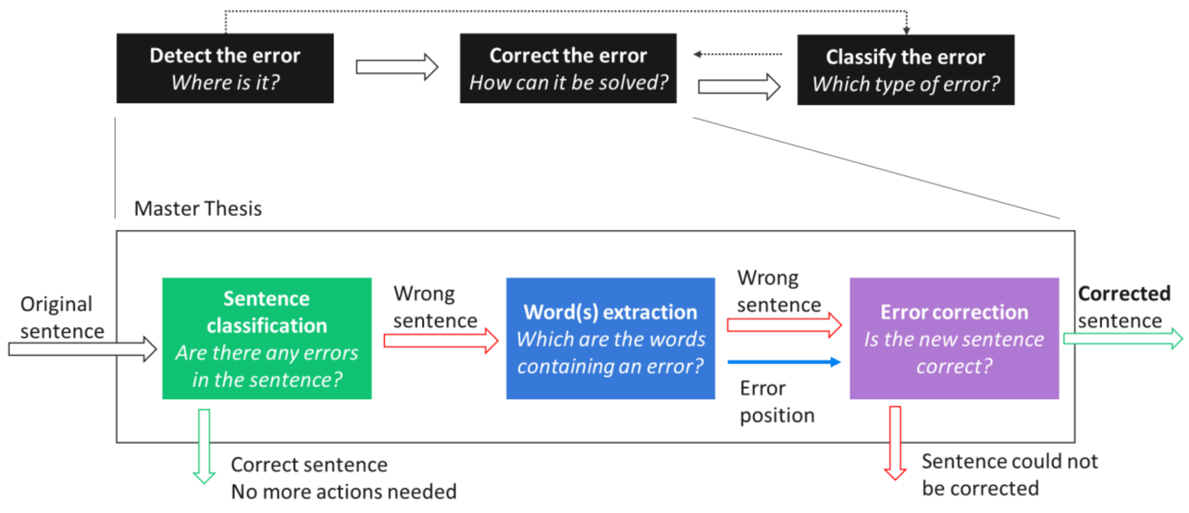


Figure 8 Master Thesis structure within the error-classification problem context

- Choose an adequate programming language. When choosing a programming language, some aspects must be considered, such as run-time speed, expressiveness, reliability, or available libraries.
- Consider testing all the time. The intrinsic modularity of the process has been a good guideline at the time of deciding which tests to conduct. Tests have covered both the learning of the modules and their performance. Besides, some naïve solutions have been tested to identify the main challenges of grammatical error detection.

Alternative research strategies were also considered. Elio et al. (2011) mention formal, experimental or process methodologies, among others. Formal methodologies are used to prove facts about an algorithm or deep dive on it. In our case, we have not delved into why any of the tools that we use work in that way, or in their theoretical model, and therefore it was not a suitable strategy. Experimental methodologies are mostly related to test new computer science solutions. A system similar to ours was not yet in place; therefore, a research strategy just dedicated to test it was not completely suitable. Finally, process methodologies are usually employed when studying the interaction between users and software solutions, typically involving human subjects. Such a study was not aligned with the goals of this work.

Grounded theory is another typical empirical research strategy. However, unlike experiments, where a theory is already formed, grounded theory starts with the data and builds theories on it (Johannesson & Perjons, 2014). As such, it is well suited to qualitative, exploratory research and the exploration of new ideas (Denscombe, 2014). One of the key weaknesses of grounded theory is that its findings are not generalizable. Given that weakness, and that its focus is on generating theories rather than testing them, we have not chosen it for our study.

3.2 Execution environment

As a first step, we have settled the conditions for our development environment, i.e. which common tools and sites we were going to use. In terms of the programming language, we have used Python, which is the current standard for machine learning and data science applications. We have also used many of the widely available libraries and packages for data science and machine learning, such as TensorFlow[6] and SciKit-learn.[7]

The code for the experiments, as well as the execution environment, has been done using Google Colab.[8] Google Colab is a platform that allows running Python notebooks on the cloud, using GPUs as hardware accelerators and enabling sharing of results, thus fostering collaboration. Each running notebook (or session) allocates 12.72 GB of RAM memory and 68.40 GB of disk space on a virtual machine, which are freed once the session finishes. The notebooks are stored in Google Drive.

[6] *https://www.tensorflow.org/*

[7] *https://scikit-learn.org/stable/*

[8] *https://colab.research.google.com*

3.3 Corpus selection and datasets

One key aspect of grammatical error tasks is the corpus selection, i.e. the set of sentences or words that constitute the dataset. Testing the different models with the same corpora is the only way to be able to compare the results of the different systems, not only in this work but also in the literature. That is the main reason why all published papers explicitly discuss the corpora as a fundamental part of the research context, and why there is a list of standardized corpora publicly available for NLP tasks.

The corpus used throughout the experiments was the FCE dataset (discussed in more details below), mainly because it is predominant in the literature in recent years, as discussed in Chapter 2. However, it is not advisable to jump directly on to the FCE dataset when developing the system, as it is quite complex. To verify and troubleshoot the code during the work, we have used so-called "toy datasets", which might be defined as fictional datasets for exploratory analysis. They should be both relatively small yet still large enough to train a model, and well-balanced (Fabisiak, 2019).

3.3.1 FCE v2.1 Dataset

The First Certificate in English (FCE) dataset (Yannakoudakis et al., 2011) is a subset of the Cambridge Learner Corpus (CLC) containing sentences extracted from 1,244 written answers produced by students taking the upper-intermediate level exam.

In this thesis work, we have used the FCE v2.1 dataset,[9] which is a modified version of the original FCE corpus made available by Yannakoudakis et al (2011). FCE v2.1 was released for the BEA-2019 Shared Task on GEC (Bryant, Felice, Andersen, & Briscoe, 2019) and was standardized with ERRANT error types and tokenized with spaCy,[10] a popular software library for NLP in Python.

The dataset consists of three data files for *train*, *test* and *development* sets respectively, each containing golden annotations by experts. Each file is composed of text sentences, and each sentence contains one or more annotations. The dataset is in M2 format (the standard format for annotated files used in GEC shared tasks). In this format, each original sentence is denoted by "S"; while edit annotation lines are denoted by "A". Edit annotation lines consist of start and end token offsets for the correction, the ERRANT error type, and the tokenized correction string. There are two other additional data fields that are not needed for our analysis and are thus ignored.

It is important to mention here that sentences containing a "noop" annotation were deemed to be correct by the annotators and thus did not require any corrections. An example of such a sentence can be seen in Figure 9.

```
S Both of us are keen on pop music .
A -1 -1|||noop|||-NONE-|||REQUIRED|||-NONE-|||0
```

Figure 9 Sample sentence from FCE dataset with no correction

Figure 10 shows an example of a sentence from the train set in M2 format, along with its ERRANT-standardized annotation and error types. The first annotation edit replaces the tokens "Because of" with "For". The second replaces "reason" with "reasons" and the third adds the token "an" preceding "spent". The resulting sentence thus reads: "For the reasons above, I, indeed, spent an awful last night in London."

[9] *Available at https://www.cl.cam.ac.uk/research/nl/bea2019st/data/fce_v2.1.bea19.tar.gz*

[10] *https://spacy.io*

```
S Because of the reason above , I , indeed , spent awful last night in London .
A 0 2|||R:PREP|||For|||REQUIRED|||-NONE-|||0
A 3 4|||R:NOUN:NUM|||reasons|||REQUIRED|||-NONE-|||0
A 11 11|||M:DET|||an|||REQUIRED|||-NONE-|||0
```

```
~~Because of~~ the ~~reason~~  above , I , indeed , spent an awful last night in London .
    0     1  2    3        4    5 6 7     8    9   10       11    12    13  14   15    16
For         the reasons above , I , indeed , spent an awful last night in London .
```

Figure 10 Sample sentence from FCE dataset and correction

Table 4 below details the number of unique sentences per file, as well as class distribution. Since we will be training our model to do binary sentence-level classification, we interested in two classes, namely 0 (grammatically correct) and 1 (grammatically incorrect, or erroneous). The class of a given sentence is determined by its annotations: if the only annotation a sentence has is "noop", that sentence is correct. Otherwise, it is erroneous (regardless of its specific annotations or their number).

Table 4 – FCE data files metrics

FCE file	Unique sentences (N)	Binary classification			
		0 (correct)		1 (erroneous)	
		N	%	N	%
train	25622	7867	31%	17755	69%
dev	2056	672	33%	1384	67%
test	2494	692	28%	1802	72%

3.3.2 Artificial Errors

Before moving on to the FCE corpus, we also created our own toy datasets by introducing errors in sentences to create a set of data to work with. The main advantage of that is that we have been able to introduce specific mistakes in the sentences, progressively increasing the complexity, to evaluate the kinds of mistakes the system worked better with. Some examples of introduced mistakes are:

- **Scramble**: Take any random word and scramble its letters, thus producing an inexistent, invalid word. This is the simplest kind of error, because the new word is unlikely to be in any dictionary. A small amount of noise is expected, since words can be scrambled in a way that still results in a valid word, such as the transformation *saw* → *was.*
- **Add-s**: Take any word and add an *-s* at the end. This error is more difficult to detect, because the resulting new words might be valid and meaningful in certain contexts, thus generating noise. An example of this phenomenon is the transformation *toward* → *towards*, which is still a valid word.
- **Preposition-swap**: Exchanging specific prepositions with others that are valid only in very specific contexts. A good example of this would be the exchanging the preposition *in* for *minus*. This type of transformation produced much less noise.

Finally, all datasets have been prepared to adequate them to the tests, for example by eliminating the repeated sentences.

3.4 System development and evaluation

Now that the research strategy and the tools are clear, including the corpora and datasets, it is time to describe the development of the system itself, as well as the evaluation performed on it.

Figure 11 incorporates the proposed models to the system diagram of Figure 8. For the first Module, two implementations are proposed, evaluated and compared: a RF (Random Forest) model and a BERT implementation fine-tuned for the purpose.

The output of Module 1, if the sentence is classified to contain at least one error, is relayed to the second Module. In Module 2, both the explainability tool LIME as well as the fine-tuned BERT's attention heads are evaluated and compared. The output of this Module should be the position of the words that are tentatively causing the sentence to be incorrect.

Finally, both the original (incorrect) sentence from Module 1 and the position of the erroneous word are relayed to Module 3. With this information in place, the dedicated BERT for masked language model can be used to obtain counterfactuals to suggest changes in the highlighted positions that would make the sentence correct.

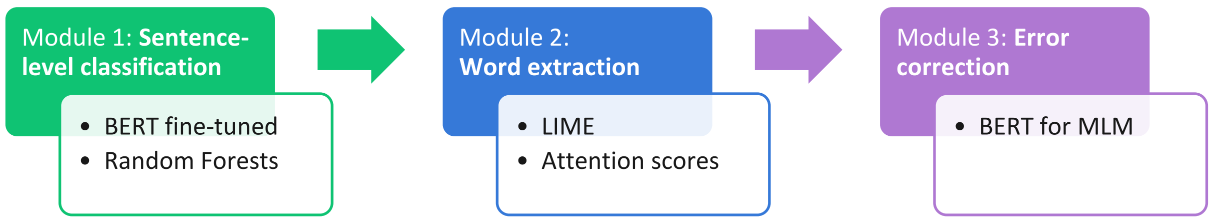

Module 1: Sentence-level classification
BERT fine-tuned
Random Forests
Module 2: Word extraction
LIME
Attention scores
Module 3: Error correction
BERT for MLM

Figure 11 Proposed models to implement each Module of the error detection system.

3.4.1 BERT start-up

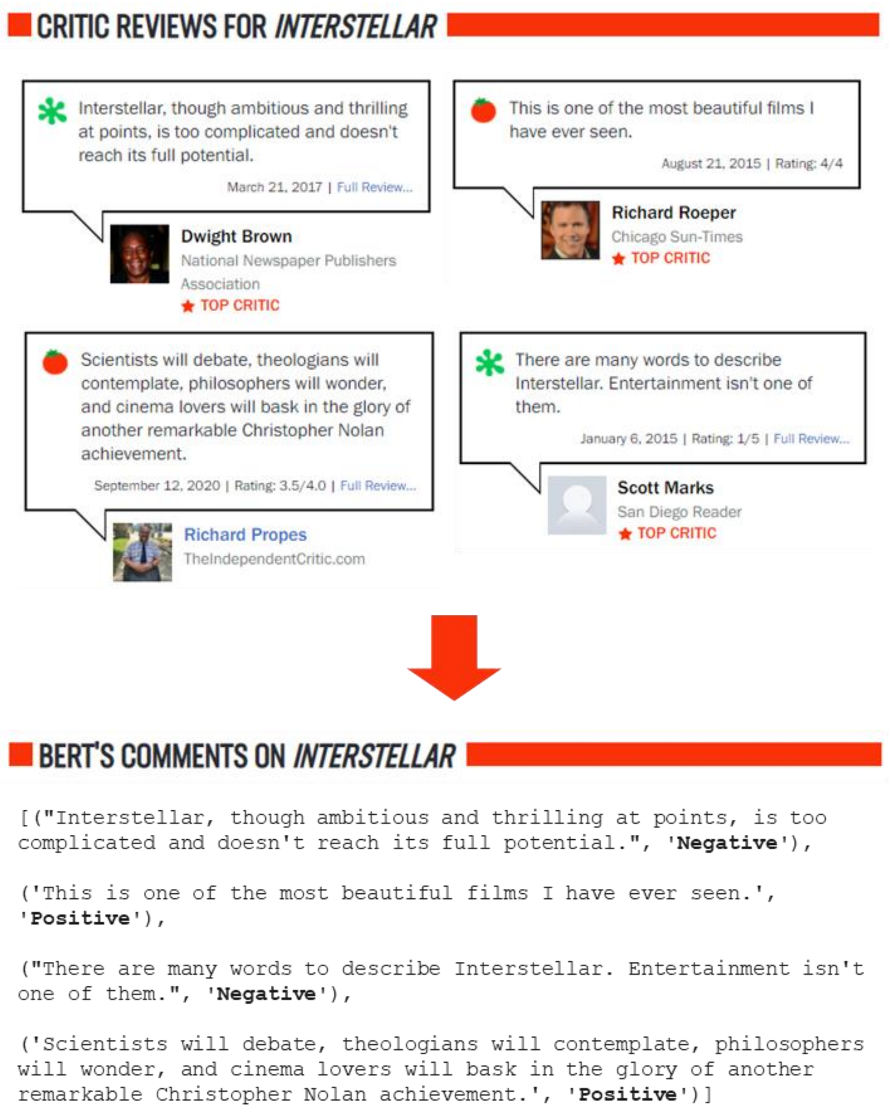
As a first approach to BERT, and to gain understanding about it, we implemented a sentiment classifier. In this case, BERT is fine-tuned with a dataset of movie reviews ("Text Classification with Movie Reviews," n.d.), and after that it is able to identify whether a review is positive or negative.

This is one common entry point to BERT, as proven by the number of examples found in the scope of this work[11]. Figure 12 represents the output of such a classifier. The critics are real comments on the movie "Interstellar" obtained from the Rotten Tomatoes website.[12]

It is very useful to start with this initialization before jumping into error tasks. In fact, the review classification is another type of sentence-level classification. Once we learned how to use and configure BERT, we could move on to our real problem.

[11] *https://github.com/google-research/bert/blob/master/predicting_movie_reviews_with_bert_on_tf_hub.ipynb*

[12] *https://www.rottentomatoes.com/m/interstellar_2014*


CRITIC REVIEWS FOR *INTERSTELLAR*

Interstellar, though ambitious and thrilling at points, is too complicated and doesn't reach its full potential.
March 21, 2017 | Full Review...
Dwight Brown
National Newspaper Publishers Association
★ TOP CRITIC

This is one of the most beautiful films I have ever seen.
August 21, 2015 | Rating: 4/4
Richard Roeper
Chicago Sun-Times
★ TOP CRITIC

Scientists will debate, theologians will contemplate, philosophers will wonder, and cinema lovers will bask in the glory of another remarkable Christopher Nolan achievement.
September 12, 2020 | Rating: 3.5/4.0 | Full Review...
Richard Propes
TheIndependentCritic.com

There are many words to describe Interstellar. Entertainment isn't one of them.
January 6, 2015 | Rating: 1/5 | Full Review...
Scott Marks
San Diego Reader
★ TOP CRITIC

BERT'S COMMENTS ON *INTERSTELLAR*

```
[("Interstellar, though ambitious and thrilling at points, is too
complicated and doesn't reach its full potential.", 'Negative'),

('This is one of the most beautiful films I have ever seen.',
'Positive'),

("There are many words to describe Interstellar. Entertainment isn't
one of them.", 'Negative'),

('Scientists will debate, theologians will contemplate, philosophers
will wonder, and cinema lovers will bask in the glory of another
remarkable Christopher Nolan achievement.', 'Positive')]
```

Figure 12 Examples of predictions made by the BERT-based sentiment classifier.

3.4.2 Module 1: Sentence-level classification

The first Module of the system covers the implementation of a binary problem: does the input sentence contain an error or not? If the answer is no, then the sentence is identified as correct and no more actions are needed. Otherwise, the Module 2 is used to locate where the error is.

The implementation can be done using both BERT and RF. The former has been proved to exhibit good performance, yet it is tailor-made specifically for NLP tasks. RF presents the advantage of versatility, since it is a model used in all fields of data science, not only language-related tasks. As such, it is considered to be our baseline classifier. That is a good reason to assess the performance of both models, to evaluate the trade-off of performance vs. complexity.

As a first step, we evaluated which hyperparameters were most suited for the task. Examples of these for BERT are whether to use ReLU or GELU as activation functions; batch size; learning rate; among others. For RF, the following parameters were studied: criterion (gini or entropy) and number of estimators.

Once the best hyperparameters were decided, a learning curve was constructed for each. This is of utmost relevance on this first Module since the detection at a sentence-level should be the simplest task of the whole system and any deficiencies here would likely only be exacerbated further downstream. In our case, we obtained the learning curves as a function of the number of training sentences and compared against the development set that remained constant throughout the experiment. For each training set, we evaluated the performance of the model for both the training sentences and the test sentences. Each result was then plotted in a chart. The chart was then used as a diagnose tool to determine whether the model is overfitting or underfitting, as well as the threshold after which adding more training samples does not enhance the results.

For the evaluation of both performance and learning curve, cross-validation scores have been used. Examples of such scores include accuracy or AUC.

3.4.3 Module 2: Word extraction

If a sentence is classified as incorrect, then the second Module must determine which word causes or contributes to the mistake. As in the previous Module, there are several ways to implement such a process. One of them is to fine-tune BERT in a certain way so that the output tokens are tagged with a label that determines if the token is likely to be the cause of the error, or not. This has already been studied in the literature in the work of Schmaltz (2019).

However, there is still a lack of works that explain whether the approach called "model explainability" would be an effective solution for this Module. In this sense, LIME - introduced in Chapter 2 - can come into play. In the evaluation of the second Module, we study the potential of LIME as a model explainability tool, also making use of cross validation scores.

Besides LIME, we also look into whether BERT's attention scores would be a suitable model explainability method. Clark et al. (2019) have demonstrated in their influential paper that lower (i.e. deeper) layers attend to more basic aspects of language, while higher layers are able to attend to more sophisticated aspects. Furthermore, certain heads attend to specific aspects of the language (e.g. certain heads attend more to prepositions, etc.). This would suggest that the scores from the attention heads could be used for word extraction and model explainability.

Other recent studies on BERT have been also focusing on whether BERT's attention heads could be used for mode explainability with arguments in favor (e.g. Vashishth, Upadhyay, Tomar & Faruqui, 2019; Wiegreffe & Pinter, 2019), and against (Jain & Wallace, 2019). In favor of attention as a suitable explainability mechanism that could be employed in this Module, is fact that there are heads that attend to certain linguistic notions, with some heads attending to prepositions, others to objects of verbs, and so forth, as demonstrated by Clark et al. (2019). Against it, is the fact that attention weights are presumably uncorrelated with gradient-based measures of feature importance, and it is possible to derive different attention distributions yielding similar or equivalent predictions, as demonstrated by Jain & Wallace (2019).

This suggests that we could "fine-tune" the word extraction Module to use the best layer and head combination **overall** for our data and then use that combination to extract words from unseen data. Given a ground-truth set of words to be extracted, one could use the layer/head combination that offers the *best* results. This, however, is not case for real-world scenarios, where the system no access to a ground truth. In other words: the decision of which layer/head combination to use cannot be decided dynamically on-demand. Thus, using the combination with the best results overall from this fine-tuning phase offers us a good compromise, as will be discussed later in chapter 4.3.2.

Just as was done in Module 1, the work done in this part could then be further subdivided into finding the best hyperparameters using the development set, followed by an evaluation based on both the development set and the actual test set. Examples of those parameters in the case of LIME are: kernel width, maximum number of samples, feature selection, among others. For the attention heads, it was just a matter of selecting the layer/head combination yielding the best results.

The evaluation was carried out using Top N Accuracy.

3.4.4 Module 3: Counterfactual-based correction

Finally, this Module is in charge of correcting the sentence based on location of the error in the original sentence. In practice, this Module will be made up of two components: Candidate Generation and Counterfactuals.

The Candidate Generation component is responsible for generating suitable candidates for counterfactuals and that could be used to solve the error pinpointed by Module 2. As such, it takes as input the index of the erroneous token that was determined by that Module and uses it to produce suggestions that could potentially correct the original incorrect sentence. This component is not concerned with verifying whether the candidates it generates would actually result in a grammatically correct sentence, should they be substituted into the original erroneous text, as that is the responsibility of the next component. Hence, its output consists of one or more tokens (i.e. the candidates) that typically have a similar function as the erroneous token. For example, if the erroneous token is an adjective, the output of this component will likely (though not always) be a list of adjectives.

Seeing that BERT has been pre-trained on large corpora, the actual implementation of the first component will be done by taking advantage of BERT's transfer learning capabilities to perform Masked Language Modelling to predict masked words in a sentence. MLM is the task of "decoding" a masked token in a sentence. Decoding here simply means "obtaining" candidates to replace the masked token. In the context of this thesis, this is essentially a simple and intuitive counterfactuals method that we can use to effectively achieve grammatical error correction.

Finally, the counterfactuals component is responsible for making sure that the candidate words from the previous component can be used in the original sentence to correct the error and thus be considered as an actual counterfactual. In practice, this means that we categorize the correction to be counterfactual-based as it returns a sentence that is classified to be correct by the sentence-level classifier in Module 1.

Since this module did not require any fine-tuning process, we could move directly to the evaluation phase. It was carried out using Accuracy, MRR, and manual validation via linguistic annotators.

3.5 Validity of Ground Truth

When evaluating the performance of a GEC system, one must compare the system outputs (i.e. the corrections) against explicitly labeled gold-standard references. However, seeing that GEC is a subjective and relatively poor-defined task (Napoles et al., 2015), it is often not straight-forward. Unlike Machine Translation tasks in which a source sentence is often completely changed, GEC often requires only small parts of a sentence to be rewritten. Additionally, an erroneous sentence can often be corrected in completely different – yet valid – ways. It is also the case that some source sentences do not require corrections at all, as they were already written correctly.

Consequently, these difficulties are reflected in the field of inter-annotator agreement, or lack thereof: when the same source sentence is analyzed and annotated by experts, they very often do not reach the same conclusions. This can be attested to by the multitude of papers written on the subject of inter-annotator agreement and ground truth and the challenges therein (e.g. Felice and Briscoe, 2015; Choshen and Abend, 2018; Bryant and Ng, 2015). This can be mitigated to some degree, given that system performance is done on the annotations of a single expert, thereby eliminating the issue of lack of inter-annotator agreement. However, this introduces a new issue: the system is no longer evaluated solely on grammatical acceptability alone, becoming, instead, “constrained by the idiosyncrasies of particular annotators” (Bryant & Ng, 2015, p. 1) .

As an example of the challenges in obtaining a good ground truth set, consider the error types from ERRANT. An analysis conducted by Korre et al. (2020) on the outputs produced by this tool for the FCE data file demonstrates that the most frequent error type was R:OTHER. Since ERRANT is a rule-based system, this means that errors of this type did not fall into any other category (Bryant et al., 2017). Further, by sampling the first 600 sentences of the FCE train data file along with the output by ERRANT and then manually re-labelling them, Korre et al. (2020) demonstrated that, despite ERRANT’s strengths, roughly 40% of the R:OTHER annotations should be re-classified.

Nevertheless, despite the challenges, human evaluation and input is often said to be the ultimate judge of quality in NLP tasks (Birch, Abend, Bojar, & Haddow, 2016) . Hence, human input is required for proper system evaluation. These factors should be taken into account and should always be considered when evaluating GEC systems and their results.

3.6 Research Quality

When planning the research, the following items were considered: validity, reliability, generalizability, and objectivity.

According to Denscombe, **validity** “refers to the accuracy and precision of the data” (2014, p. 271), as well as appropriateness to the research question. Throughout this work, we took care that the research question was valid and formed based on existing theories and the literature on the topic of GEC. We have also taken care that the experiments designed to investigate the research question were in line with the overall goals of this study. For each Module developed, we have chosen appropriate measures that meaningfully evaluated the component in question. Furthermore, as an additional step towards research validity, our results have been evaluated by a professional linguist. The results of that evaluation can be found in Appendix E.

Nevertheless, when assessing validity in NLP research, one must always consider the difficulties and challenges with ground truths and inter-annotator agreement discussed in chapter 3.5.

Reliability refers to the consistency of results of the research instruments, as well as their neutrality (Denscombe, 2014). When designing the experiments, we sought to employ the most appropriate and the most suitable hyperparameters for each model developed, while respecting the established methodology for machine learning projects and providing details for our design decisions. Additionally, we employed a publicly available dataset. By those measures, as well as by making sure that the dataset used in the final evaluation was not the same employed during training and validation, we ensure that the results are reliable.

Generalizability has to do with external validity of the data analysis and results (Denscombe, 2014). Regarding this point, we must advise caution and must concede that the number of samples used for validation of the experiments is relatively low. However, we have used well-known components that are recognized as state-of-the-art in their respective spheres of influence.

We start with LIME. Given a proper fine-tuning of the underlying “black-model” classifier on which it is evaluated, as well as using the hyper-parameter configurations described in this study, we expect it to generalize well to similar data. As for BERT, it has quickly become the state-of-the-art model in NLP tasks. As discussed earlier, we have used the public FCE dataset to fine-tune BERT. All these serve to reinforce the generalizability of this study.

Denscombe makes a statement concerning **objectivity** that no research can ever be said to be free of the “influence of those who conduct it” (2014, p. 319). While this is true for both qualitative and quantitative research, it is more so of qualitative data. Quantitative research – as opposed to qualitative research – has the advantage of being more grounded and, therefore, more objective. Despite this research being based on quantitative data research methods and analysis, we have sought to address this concern by motivating our choices of methods and parameters, so that the reader can judge for themselves on this criterion. Here it is worth repeating that our final results have been evaluated by a professional linguist, which contributes to enhance this research’s objectivity factor.

3.7 Research Ethics

This work does not involve social research and no individual data has been collected. Nevertheless, the discussion around ethics can be expanded to address potential issues augmented by data science. In 2017, a report by the French Data Protection Authority CNIL (Commission Nationale de l’Informatique et des Libertés) outlined six main ethical concerns in data science (“Algorithms and artificial intelligence: CNIL’s report on the ethical issues,” 2018). Two of those issues are of particular interest in the context of this work and could potentially be a concern for this present work as well:

- **Algorithmic bias**: “systematic and repeatable errors in a computer system that create unfair outcomes, such as privileging one arbitrary group of users over others” ("Algorithmic bias," n.d.).
- **Algorithmic profiling**: “purposeful recording and classification of data related to individuals” which is usually the result of an automated data mining process (Büchi et al., 2020).

For our research we have made use of publicly available datasets, absent of any information allowing the identification of individuals – on either annotators or students – or any means of tracing back to particular persons. However, as is the case with most data science projects, there is always the potential that algorithms produced as a result of this thesis project might be misused, resulting in the issues outlined above. In GEC applications that could happen, for example, if an individual’s performance is tracked and evaluated in terms of their grammatical prowess and performance, although such a scenario seems farfetched and is only mentioned for the sake of transparency and completion.

4 Results and discussion

4.1 Preprocessing of datasets

As explained in 3.3.1, we would be using the FCE dataset to run our experiments. Nevertheless, some preprocessing needed to be done in order to be able to make use of the dataset for our purposes.

The class labels are defined as follows. The negative class (“0”) is assigned to correct sentences and hence do not need to be corrected. The positive class (“1”) is therefore assigned to sentences with one or more grammatical errors.

4.1.1 Training set

The FCE training set consists of 25,622 unique sentences. Each sentence may be correct (no grammatical error), or incorrect (with one or more grammatical errors). Nevertheless, there are capitalization errors that a case-insensitive model like ours cannot detect. Consequently, the first round would be to remove sentences whose correction is the same word though capitalized differently. Due to that, 184 sentences needed to be removed, resulting in the training dataset being reduced to 25,438 sentences.

Looking at the class distribution, the 25,438 sentences are distributed between 17,571 that belong to the positive class and 7,867 corresponding to the negative class. Since this dataset is going to be used for training, a balanced dataset is desired. Here the strategy was to replicate the sentences belonging to the negative class until a more balanced dataset was obtained. This resulted in a total of 33,000 samples: the original 17,571 belonging to the positive class, and 15,429 sentences corresponding to the negative class. This process is depicted in Figure 13.

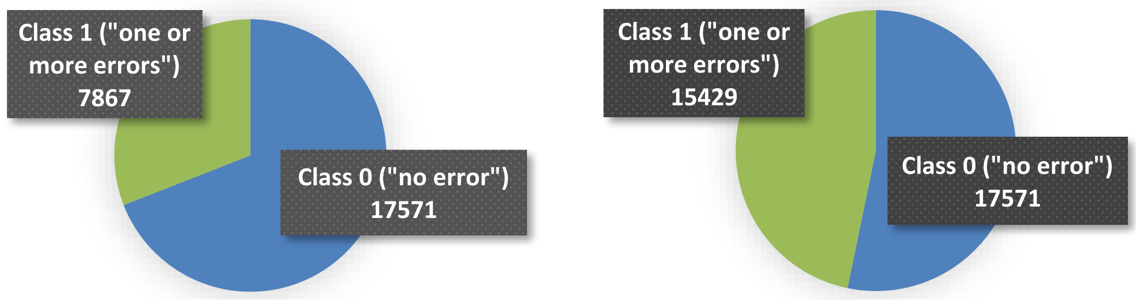


Figure 13 Class distribution of the train dataset before and after replication

4.1.2 Development set

The FCE development set consists of 2,056 unique sentences. Again, due to the fact that capitalization errors cannot be detected, 17 sentences needed to be removed, for a total of 2,039 sentences.

The development set will be used throughout the experiments; therefore two variants need to be created. The first one is the development set that will be used in Module 1 for model validation. No more adaptations needed to be done and we could use all 2,039 sentences for this purpose. The class distribution is as follows: 1,367 sentences corresponding to the positive class, and 672 sentences corresponding to the negative class.

The second variant is used for Modules 2 and 3. This work is constrained to sentences with only one incorrect word per sentence, as explained in chapter 1.4. This means that, for Modules 2 and 3, we should only consider sentences with one error word since the word extraction and the counterfactual will be based on that erroneous word. The number of sentences falling in this category is 273, all of them belonging to the positive class. This will be referred to as the single-error development set.

The class distribution of the development dataset is shown in Figure 14.

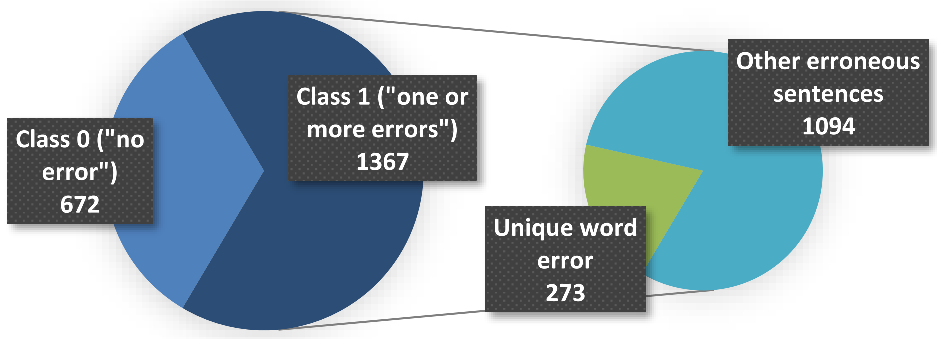


Figure 14 Class distribution of the development dataset

4.1.3 Test set

The FCE test set consists of 2,494 unique sentences. This time, 25 sentences needed to be removed, for a total of 2,469 sentences.

The test set will be used to validate the training and fine-tune results, therefore the same two variants as for the development set will be created. For Module 1, nothing else needs to be done and we could use all 2,469 sentences for this purpose. The class distribution is as follows: 1,777 sentences corresponding to the positive class, and 692 sentences corresponding to the negative class.

As already explained, for Modules 2 and 3 we will only consider sentences with one error word. The number of sentences falling in this category is 323, all of them belonging to the positive class. This will be referred to as the single-error test set.

The whole class distribution of the test dataset is shown in Figure 15.

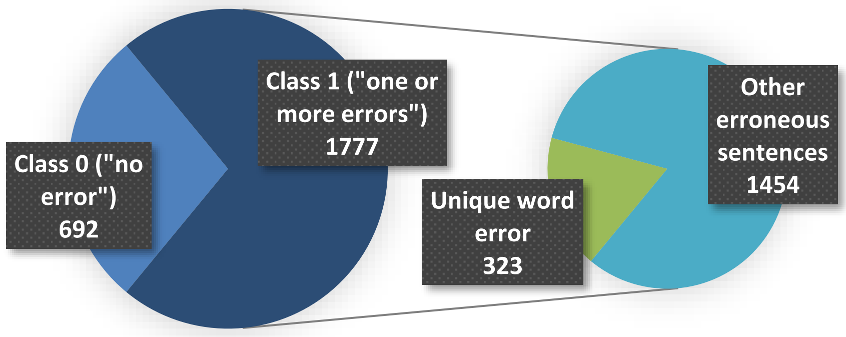


Figure 15 Class distribution of the test dataset

4.2 Module 1: Sentence-level classification

As explained in Chapter 3.4.2, this Module instantiates a binary classifier on the sentence level, whose indicates whether a sentence is correct or incorrect. Two models are considered: BERT and RF.

When evaluating the models used in Module 1, learning curves were drawn for each model. The procedure was the same throughout and is described in more details next. The learning curves themselves will be presented in sub-chapters 4.2.2 and 4.2.1.

The training dataset was partitioned into divisions, each containing, respectively, 500, 1000, 2000, 4125, 8250, 16500, 24750, and 33000 sentences; with each division containing all the sentences from the smaller subsets. This means that, for example, the training split with 1000 samples contained the 500 sentences from the first training split plus 500 new samples.

Each training split was validated on the whole development dataset, and AUC scores were obtained for both the corresponding training split and the development dataset.

4.2.1 BERT results

Before getting into BERT fine-tuning, we needed to decide between three different BERT implementations to instantiate: the official version of BERT[13], stored in the original repository; the one created by HuggingFace[14] which, according to BERT's authors, performs as well as the original BERT; and the one by ktrain[15], which internally instantiates the model by Hugging Face, while encapsulating most of the details to make development faster.

We decided to use the version developed by Hugging Face, due to two reasons. Firstly, the official BERT used an old version of TensorFlow, which could interfere later on in the development process. Secondly, ktrain, although it provided a nice wrapper for everything we needed, restricted the customization required for this work as we needed to get information on the internals (e.g., attention scores) of each BERT layer later on in this work.

Next, we needed to decide the architecture to put on top of BERT's output to be the sentence level classifier based on the information carried on the [CLS] token. Devlin et al. (2018) indicates that ReLU and GELU are good candidates for activation functions on the output layer, and Hugging Face suggests tanh. Other parameters to be fine-tuned are: batch size (16 or 32), number of epochs (between 2 and 4), and dropout (0.05, 0.1, and 0.2). Then, a linear transformation layer is needed to reduce from the 768 hidden states to 2 possible classes (sentence correct/incorrect) and a softmax function. This is presented in Figure 16.

Besides these hyperparameters, an Adam optimization algorithm with a learning rate of 2e-5 (Devlin et al., 2018) is used for training and the negative log-likelihood as loss function in conjunction with the Log Softmax layer in the fine-tune Module.

Table 5 shows the results of different BERT architectures with varying batch size, dropout, and activation function. Accuracy and AUC scores are presented. Since the development dataset presents class imbalance, the AUC score is preferred over accuracy. The results are almost identical in all configurations, which is expected according to Devlin et al. (2018) for the cases where the training

[13] *https://github.com/google-research/bert*

[14] *https://huggingface.co/transformers/model_doc/bert.html*

[15] *https://github.com/amaiya/ktrain*

dataset is large enough. The configuration with the highest AUC was chosen, highlighted in bold in the table. An analysis of this particular configuration was tried with epochs varying from 2 to 4; nevertheless, similar results were obtained again. Therefore, we decide on the following hyper-parameters: **2 epochs, batch size of 16, probability of dropout 0.1, and tanh as activation function.**

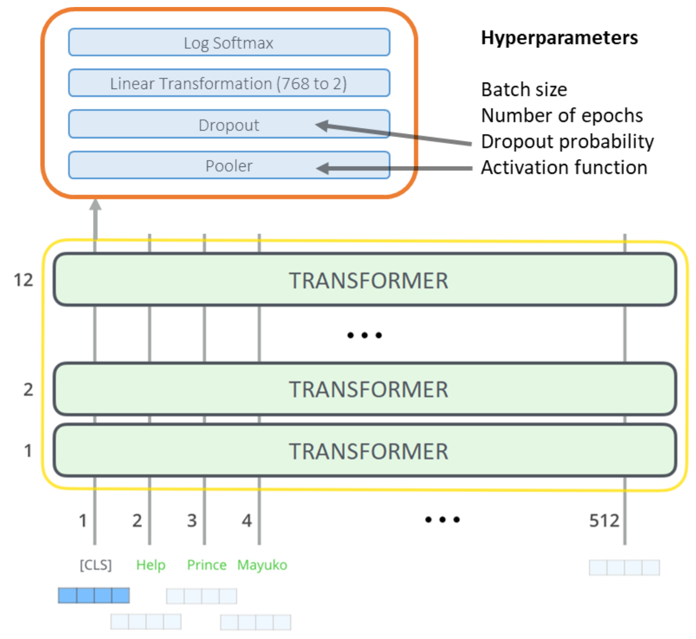


Figure 16 BERT architecture for fine tuning. The yellow Module represents BERT-Base uncased, and the red Module depicts the fine-tuning layers. Modified from (Alammar, 2018)

Table 5 – Accuracy and AUC of different BERT configurations. Trained on the full train dataset and validated on the development dataset.

Epochs	Batch size	Dropout	Activation function	Accuracy	AUC
2	16	0,05	tanh	0.821	0.834
2	16	0,05	ReLU	0.813	0.831
2	16	0,05	GELU	0.822	0.838
2	**16**	**0,1**	**tanh**	**0.818**	**0.839**
2	16	0,1	ReLU	0.819	0.836
2	16	0,1	GELU	0.808	0.831
2	16	0,2	tanh	0.816	0.833
2	16	0,2	ReLU	0.821	0.836
2	16	0,2	GELU	0.818	0.838
2	32	0,05	tanh	0.810	0.833
2	32	0,05	ReLU	0.805	0.831
2	32	0,05	GELU	0.804	0.830
2	32	0,1	tanh	0.816	0.836
2	32	0,1	ReLU	0.808	0.833
2	32	0,1	GELU	0.814	0.838
2	32	0,2	tanh	0.798	0.827
2	32	0,2	ReLU	0.814	0.833
2	32	0,2	GELU	0.808	0.832

Once the architecture was settled, the learning curve was drawn. Figure 17 plots the learning curve for the proposed BERT architecture after being fine-tuned for our purpose. The results show that after 16,500 sentences – half the training dataset – no better scores are obtained. Nevertheless, the training score drops slightly after that point, which may indicate that the model generalizes better with more samples as the validation score does not decrease.

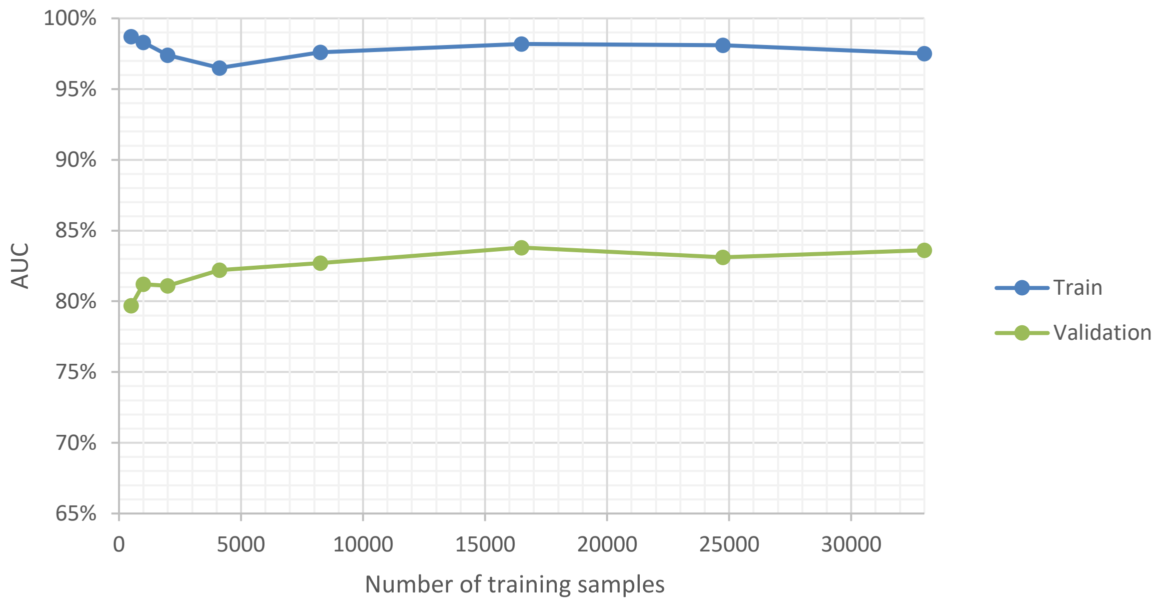


Figure 17 Learning curve for BERT. Validated on the development dataset using AUC scores.

4.2.2 RF results

For the Random Forests model, we evaluated selection criterion (*gini* and entropy) and different values for *n_estimators* (between 500 and 900; preliminary tests did not demonstrate any gains in performance with higher numbers).

Regarding tokenization, we evaluated sklearn's CountVectorizer and TF-IDF (Term Frequency Inverse Document Frequency). For TF-IDF, we evaluated the following *ngram* ranges: unigrams; unigrams and bigrams. Both tokenizers were evaluated based on lowercase conversion.

Table 11 in Appendix B shows the results of different RF models with varying number of estimators, vectorizers and, in case of TF-IDF, ngram ranges. As we have seen with BERT in the previous section, AUC is the preferred measure due to the class imbalance present in the development dataset which was used for validation. The configuration with the highest AUC (highlighted in bold in the table) was chosen: **600 estimators, 'gini' criterion and TF-IDF vectorizers with ngram_range = (1, 2)**.

As was the case with BERT, once the parameters were settled, a learning curve was obtained, which is shown in Figure 18. Again, as was the case with BERT, we notice that from around 16,500 training samples onwards the results improve only slightly, with an end AUC score of 72%.

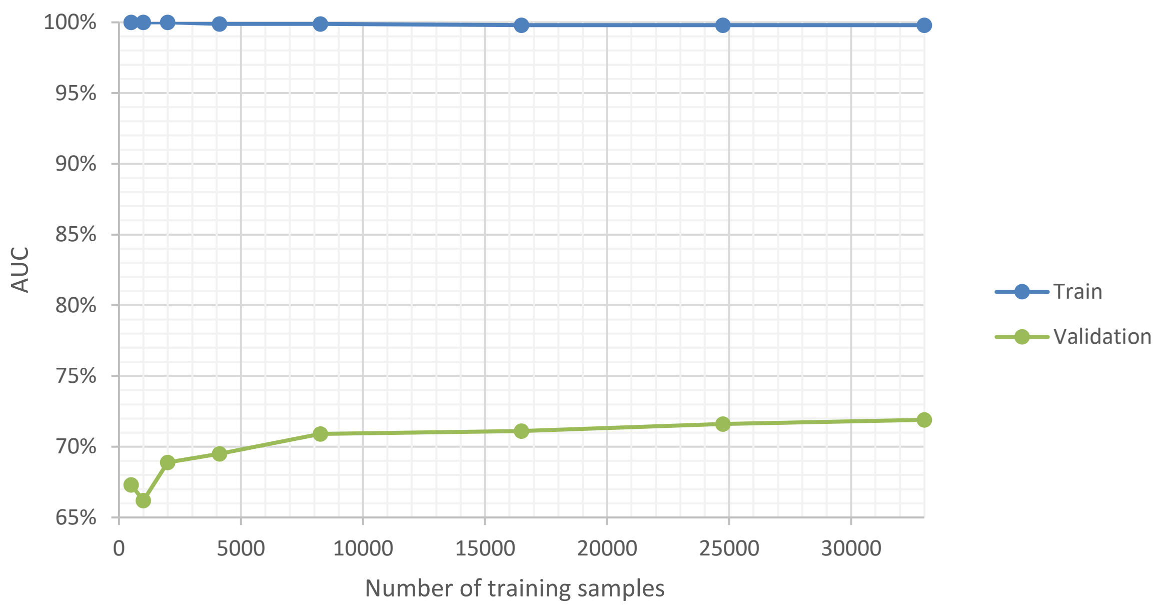


Figure 18 Learning curve for RF. Validated on the development dataset using AUC scores.

4.2.3 Discussion

With the results presented in sections 4.2.1 and 4.2.2, it is decided that we move forward with the BERT model as it shows a better AUC score (**83.9 %** vs **72 %**). This is aligned with our expectations since BERT was designed for NLP tasks, whereas RF represents a more generic model not specific to this task.

To summarize, the proposed architecture is BERT trained on the 33,000 samples of the training set, 2 epochs, batch size of 16, probability of dropout 0.1, and tanh as activation function, which obtains 83.9% AUC when evaluated in the development dataset. At this point, we saved the model weights so that the fine-tuned model could be reused for future experiments.

We run an evaluation on the test dataset to see the generalization of the model. AUC for the test set is 81.95% and accuracy is 80.4%. The full confusion matrix can be found in Table 6.

Table 6 – Confusion matrix for the test set on the proposed Module 1

		Predicted	
		Negative	Positive
Actual	Negative	591	101
	Positive	382	1395

Other experiments were run in Module 1. It is worth highlighting that we initially attempted to fine-tune BERT to do error type classification (which corresponds to the last black box in Figure 8, named “classify the error”). Nevertheless, here we were unable to obtain acceptable results: the AUC score for most of the experiments was around 50%, meaning that the model did not learn to do error type classification, likely due to an insufficient number of sample sentences per error type. Note that the training dataset consists of errors of many types, which implies that there were not so many instances of the same error type.

4.3 Module 2: Word extraction

In Module 2, the development set with single-word errors will be used to fine-tune the hyperparameters of the different models, whereas the test set with single-word errors will be used to perform a final evaluation on the models after they have been fine-tuned to test whether they generalize well.

Nevertheless, as described in 3.4.3, the input to Module 2 should be a sentence classified as erroneous by Module 1. Therefore, a preliminary step would be to pass those datasets through Module 1 to ensure that those sentences are classified as erroneous. Results of this step will be further discussed in Chapter 4.5.2.

For the single-error development dataset, out of the 273 original sentences, 108 of these were false negatives (i.e. mistakenly classified to be correct), whereas the remaining 165 are true positives (i.e. erroneous sentences classified as erroneous) by Module 1. Similarly, for the single-error test dataset, out of the 323 original sentences, 132 of these were mistakenly classified to be correct, whereas the remaining 191 are erroneous sentences classified as erroneous by Module 1.

The output produced by Module 2 consists of the index of the erroneous token in the original sentence.

As part of the development process of this Module, the artificial error datasets described in section 3.3.2 were used to gain an understanding of the different options and models. However, only results from the proper datasets will be presented.

4.3.1 LIME results

The first step to bring up LIME was to decide which hyperparameters of the LimeTextExplainer[16] component to fine-tune or change from the default ones. As it was mentioned in chapter 2.5, great care was taken to ensure that the hyperparameters for LIME were thoroughly fine-tuned so as to obtain better results. The final selection of hyperparameters for further investigation was:

- **bow** (bag of words, default True). If true, each unique word is only accounted for once, meaning that if the same word appears more than once in the sentence, they will be considered equally. Hence, we must set it to **False** for our use case since we do care about each word individually. If set to false, each word is considered as many times as it appears in the sentence.
- **split_expression** (defaults to standard delimiters such as '.', ',', ' '). The FCE dataset already comes delimited with blank spaces. Therefore, split_expression must be set to ' '.
- **kernel_width** (defaults to 25). As explained in section 2.5, the kernel width determines the vicinity area for each sample, so this parameter plays a major role. We will evaluate it from 0.1 to 100.
- **mask_string** (defaults to 'UNKWORDZ', only used if bow = False). It represents the string that will be inserted for each word that gets removed. The default value was not good for our application since adding 'UNKWORDZ' to a sentence would make it incorrect. Instead, the values we chose were '[MASK]', which is the masked token for BERT, or '', i.e., the word is removed. The latter represents the behavior when bow is true.

[16] *https://lime-ml.readthedocs.io/en/latest/lime.html*

- **feature_selection** (defaults to 'auto'). It defines which feature selection method to use. We ran analysis on 'none', 'highest_weigths', and 'forward_selection'.
- **num_samples** (defaults to 5000). As explained in section 2.5, it represents the number of masked samples to obtain per sentence. We tried 2000, 5000, and 10000.
- **Other parameters** (**num_features** and **random_state**). The parameter num_features determines how many features (or words) to perform explanations on. In our case, we set it to 5 (i.e., get a rank with the topmost 5 words). random_state represents a key parameter to set to ensure that experiments can be repeated. We set it to 1234. Also, LIME does not get reseeded every time it is called, so we needed to create a new instance before running every explanation.

Also, during the exploration phase with the artificial error datasets, we discovered that num_samples generated as many masked sentences as defined in the parameter, irrespectively of the actual length of the sentence. We realized that the total number of variations for a sentence of length l is $2^l - 2$, since each word may or may not be masked in every sample, minus the cases where either all or no words are masked away. As an example, if the sentence is "Hello world !", its length is 3 as it consists of 3 tokens. The total number of variations is $= 2^3 - 2 = 6$: "Hello" / "Hello world" / "Hello !" / "World" / "World !" / "!". Note that "Hello world !" and "" were not considered. For that, we added another hyperparameter that, when set, will limit the number of samples per sentence to be the minimum between num_samples and $2^l - 2$. We named it **constrain_samples**.

Preliminary results with the artificial errors dataset showed that the most significant parameter was **kernel_width**, obtaining the best results between 5 and 10. Similarly, other parameters yielding good results were **mask_string** = '', **feature_selection** = 'forward selection', **num_samples** = 5000, **constrain_samples** = True. Unless explicitly mentioned, these were considered to be the default parameters for the rest of the experiments.

The metrics used for comparison were Top 1, 2, and 3 Accuracy, which indicate whether the offset of the error in the original sentence matches the first, second, or third explanation by LIME.

With this in place, we ran the experiment on the single-test development dataset to fine-tune first which kernel width would work best in this case.

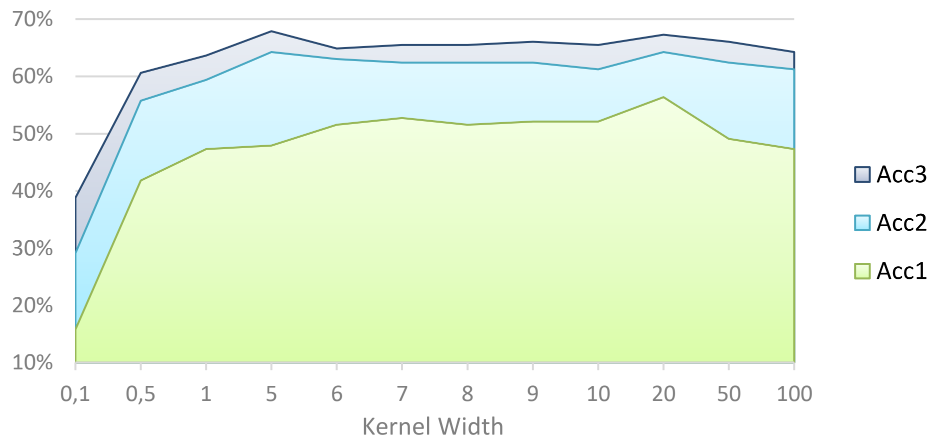


Figure 19 Results for varying kernel_width (0.1 and 100) on the single-error development set.

Results in Figure 19 show that, for the single-error development set, a better kernel width is obtained in the range between 10 and 30. Therefore, a second round of experiments was constructed, now with varying kernel width and num_samples.

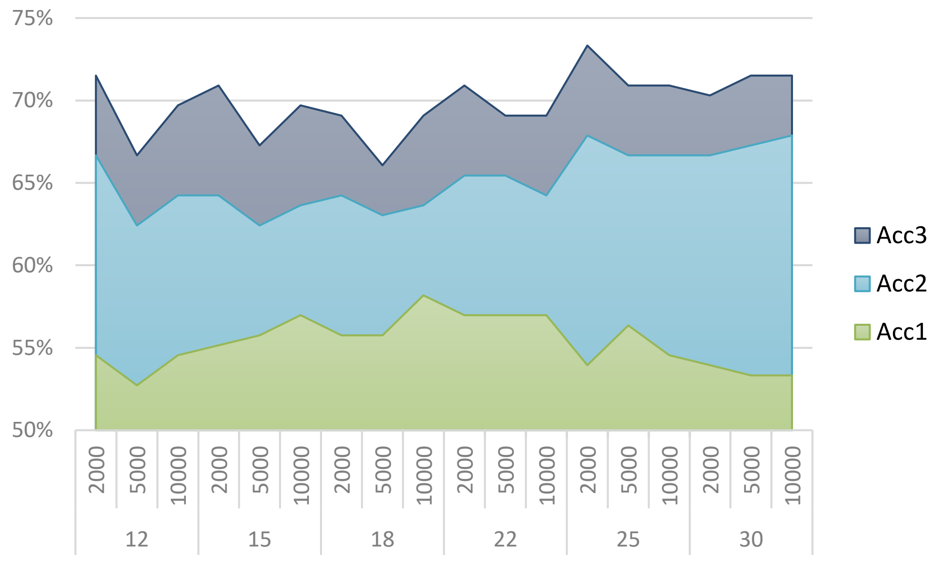


Figure 20 Results for varying kernel_width between 12 and 30 and num_samples between 2000 and 10000, on the single-error development set.

Figure 20 shows that kernel widths of 12 and 15 are always worse than the remaining widths, so we should center the analysis on kernel widths of 18, 22, 25, or 30. Regarding num_samples, there are certain combinations of kernel width and num_samples that obtain auspicious results. Those are kernel width and num_samples of 18, 10000 (highest Acc1 = 58%); 30, 10000 (highest Acc2 = 68%); 25, 2000 (highest Acc3 = 73%); and 25, 5000 (good balance in Acc1-3); respectively.

A final round of experiments was carried out on the selected kernel widths and num_samples to determine which are the best constrain_samples, mask_string, and feature_selection. The evaluation was done with one parameter at a time.

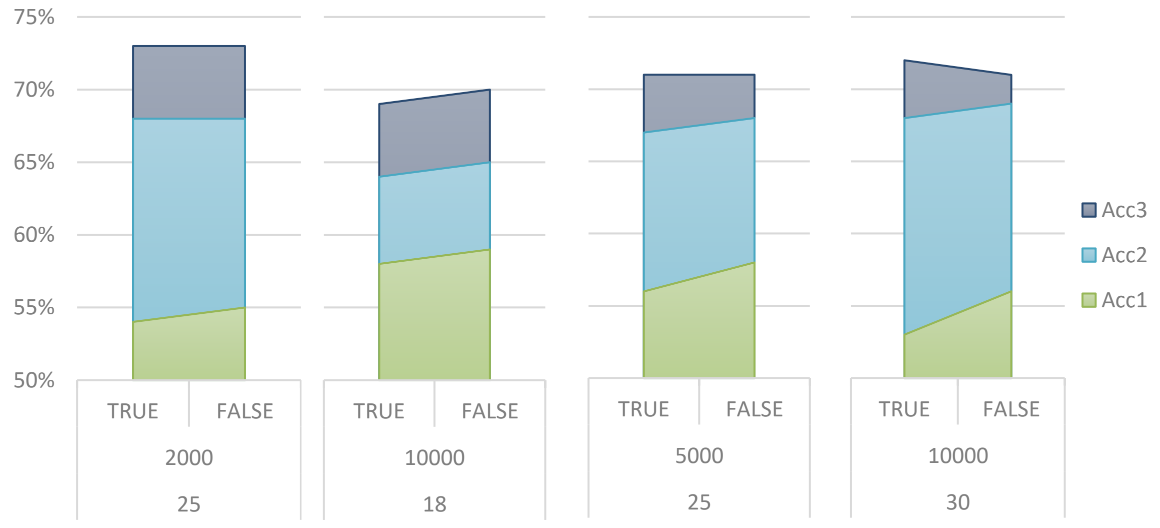


Figure 21 Results for selected kernel widths and num_samples, varying constrain_samples.

Figure 21 shows that, for the selected num_samples and kernel width combinations, the system always performs best when constrain_samples is False. Therefore, this parameter will be set to False.

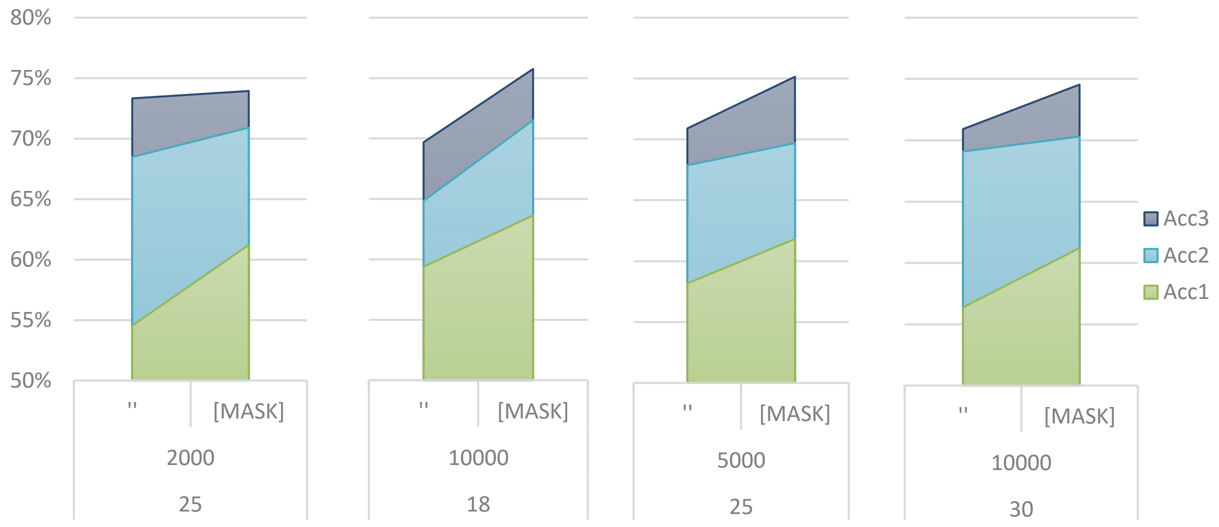


Figure 22 Results for selected kernel widths and num_samples, varying mask_string.

Similarly, Figure 22 shows that, for any of the selected num_samples and kernel width combinations (and constrain_samples = False), the system always perform best when the mask string is [MASK], i.e., the masking token for BERT. We see that the change in the scores is significant. Hence, this parameter will now be set to [MASK].

Lastly, the only parameter to be fine-tuned is feature_selection. Since it is the last test, results are presented in Table 7 for easier comprehension.

Table 7 – Final experiment in LIME with the single-error development dataset.

LIME parameters	feature_selection	Acc1	Acc2	Acc3

KW = 25
num_samples = 2000
constrain_samples = False
mask_string = [MASK]
	highest_weights	61%	67%	74%

	forward_selection	61%	71%	74%
	none	65%	75%	79%

KW = 18
num_samples = 10000
constrain_samples = False
mask_string = [MASK]
	highest_weights	64%	72%	75%

	forward_selection	64%	72%	76%
	none	65%	75%	78%
KW = 25 **num_samples = 5000** **constrain_samples = False** **mask_string = [MASK]**	highest_weights	61%	68%	75%
	forward_selection	62%	70%	75%
	none	**66%**	**74%**	**79%**

KW = 30
num_samples = 10000
constrain_samples = False
mask_string = [MASK]
	highest_weights	61%	67%	73%

	forward_selection	61%	70%	75%
	none	66%	75%	78%

With these results, we decide that the best hyperparameters for LIME are **kernel_width** = 25, **mask_string** = '[MASK]', **feature_selection** = 'none', **num_samples** = 5000, and **constrain_samples** = False, since they yield the best results in Acc1 and Acc3.

4.3.2 Attention scores results

The basic assumption behind using attention scores as an explainability mechanism is that attention scores are human interpretable, which has been positively demonstrated by Vashishth et al. (2019).

Since our model has been fine-tuned for sentence classification, the basic intuition is: use the attention scores of all the tokens in the sentence evaluated against the [CLS] token (i.e. the classification token) to determine which token carries the largest weight. That, in turn, represents the token being attended to the most by the classification token, hence influencing the classifier's prediction the most.

Although this is an intuitive idea, it becomes more complex when one takes into account the fact that there are at least 144 combinations to be evaluated (i.e. 12 heads x 12 layers in bert-base-uncased). This leads to the question: which layer and head combination to use for word extraction? As we have seen in chapter 3.4.3, the approach we have taken is to accept a compromise where the best layer/head combination during the fine-tuning phase is also used for subsequent tasks.

Another complication we encountered has to do with tokenization. As we have seen in chapter 2.3, the tokenizer used with BERT adds special characters (i.e. ##) to represent subtokens of a given word. Out-of-vocabulary words are split into subtokens, with each subtoken receiving its own attention score. Misspelled words would be particularly troublesome here, since BERT assigns a score to each and every token from the tokenized output. To solve this issue, we concatenate the subtokens back into a single reconstructed token. This is necessary, due to the fact the output of this Module consists of the index of the erroneous token in the original sentence. Thus, we concatenate the subtokens back into their original tokens to 1) preserve the original sentence's length and 2) be able to determine which token to be used as input to Module 3 (i.e. the erroneous token that will be corrected by the counterfactuals Module).

A further complication, however, is that, since each of those subtokens is assigned an attention score by BERT, they must be coalesced in some way. We evaluated three pooling mode methods to solve this challenge: **max**, **mean** and **sum**. In each method, the score determined from among a given word's subtokens is taken as the score for the reconstructed (i.e. original word). Next, the new scores are normalized for pooling modes mean and max. Finally, the index of the token with the new top 1 score is the output of this Module.

Table 8 – Comparison of pooling modes with attention scores

Sentence		***the***	***conference***	***is***	***situatet***			***here***	***.***		**Total**
Encoded	101	1996	3034	2003	26179	3686	2012	2182	1012	102	
Output	[CLS]	the	conference	is	situ	##ate	##t	here	.	[SEP]	
Attention	0.060	0.050	0.054	0.059	0.157	0.162	0.145	0.044	0.142	0.127	1.000
Sum	0.060	0.050	0.054	0.059	0.465			0.044	0.142	0.127	1.000
Max	0.060	0.050	0.054	0.059	0.162			0.044	0.142	0.127	0.697
Max_norm	0.085	0.072	0.078	0.084	0.233			0.063	0.203	0.182	1.000
Mean	0.060	0.050	0.054	0.059	0.155			0.044	0.142	0.127	0.690
Mean_norm	0.086	0.073	0.079	0.085	0.225			0.064	0.205	0.184	1.000

We illustrate these steps in Table 8. Here, the sentence "The conference is *situatet* here." contains a misspelled word "situated". That word was split into 3 subtokens by the tokenizer: "situ", "##ate" and

“##t”. This would be classified as erroneous by our fine-tuned model and we would expect the word “situatet” to “extracted” as the explanation. Note also that extra tokens – classification (i.e. [CLS]) and separator (i.e. [SEP]) – were added to the sentence by the tokenizer. The attention scores by the [CLS] token in layer 12 and head 12 are shown, along with the results for the pooling modes. In all three pooling modes the misspelled word would be “highlighted” as being the erroneous token, though pooling mode sum places more emphasis on it.

Unlike what was done when evaluating LIME, top 1 accuracy was the only measure used to evaluate performance. In case of LIME, there were parameters that needed to be tuned before performance could be finally evaluated in terms of top 1 accuracy.

Using the single-error development set as the ground-truth and to simulate a fine-tuning phase, we evaluated each method across all layer/head combinations by means of Top 1 Accuracy and found that **sum** was the most suitable pooling mode for our task. The best result was for layer 9 and head 9, as can be seen in Figure 23. The results for the other pooling modes can be found in Appendix C.

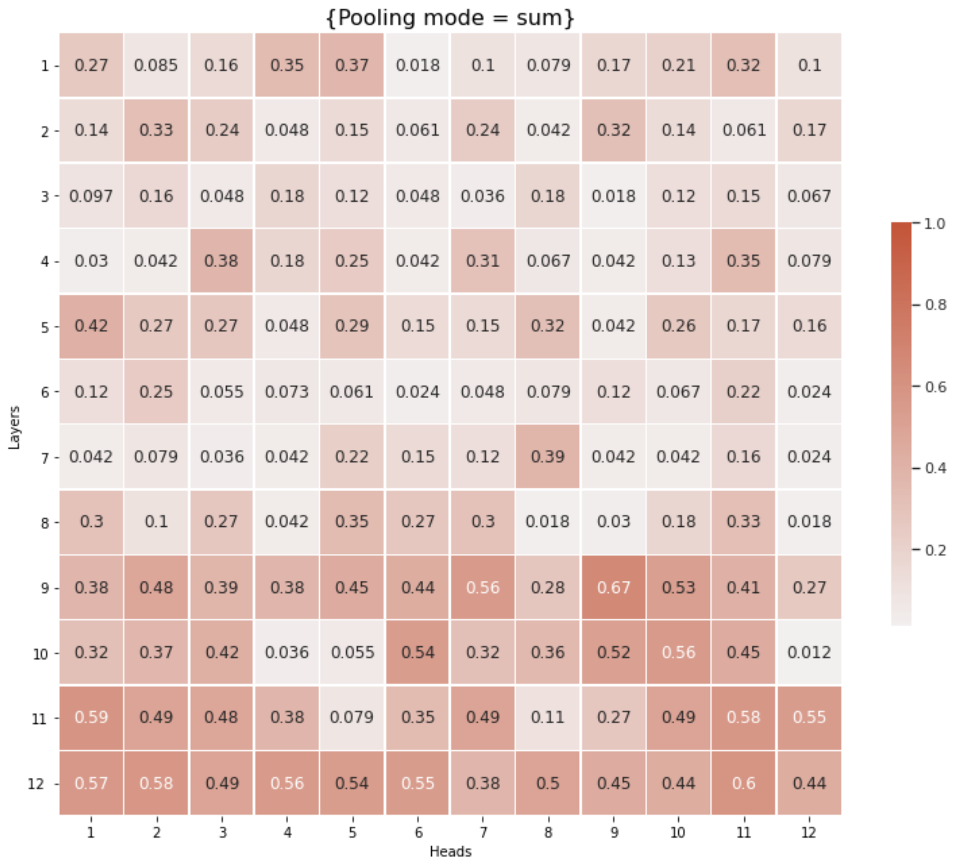


Figure 23 Top 1 Accuracy results for pooling mode "sum", for all layer/head combinations, evaluated on the single-error development set.

It is interesting to note that this combination proved to be the best one across pooling mode strategies, with results varying a little: mean achieved 64% accuracy and max achieved 65% (refer to the appendix C for more details). The fact that the single-error development set contains several different grammatical error types overall should imply that the result obtained during this “fine-tuning” phase can be expected to generalize reasonably well to similar errors in unseen data. Consequently, for all intents and purposes**, the combination for layer 9 and attention head 9, with pooling mode “sum”** is considered as being the fine-tuned setting and is chosen as the combination to be used in subsequent tests and evaluations.

4.3.3 Discussion

In order to compare which of the model performs better, another experiment was run with the decided hyperparameters on the single-error test set. For LIME, the results were Acc1 = 63%, Acc2 = 73%, and Acc3 = 77%. As compared with the results obtained for the development dataset, they have not worsened much, so we can conclude that the model generalizes well. Similarly, the same was run for Attention heads with the best layer/head combination set. Here the accuracy scores were 60% accuracy with pooling "sum", which proves that the best layer/head combination is sensitive to the dataset.

Nevertheless, it was noted that the elapsed time for both models was quite different. In the case of LIME, since a linear model needs to be trained over several permuted samples, it took 14 minutes for the whole single-error test dataset, which in average represents 4.4 seconds/sentence. This contrasts with Attention heads, where a single pass per sentence is required to get results. In this case, it took 8 seconds to go over the whole dataset, which is approximately 100 times faster. It is also worth noting that no efforts were put to optimize LIME's execution time, and some inefficiencies were found when LIME consults BERT fine-tuned to construct the linear model.

Also, to compare how both models work, a correlation analysis was done. If the results from both LIME and Attention heads are similar, the correlation will be high. The error offsets obtained by LIME and Attention on the single-error test set were compared in terms of whether the same offset was obtained. Results presented in Figure 24 show that, out of the 191 sentences, 114 sentences received a different suggestion from LIME and Attention, whereas 77 sentences both models identified the same sentence. This yields a correlation of 60%.

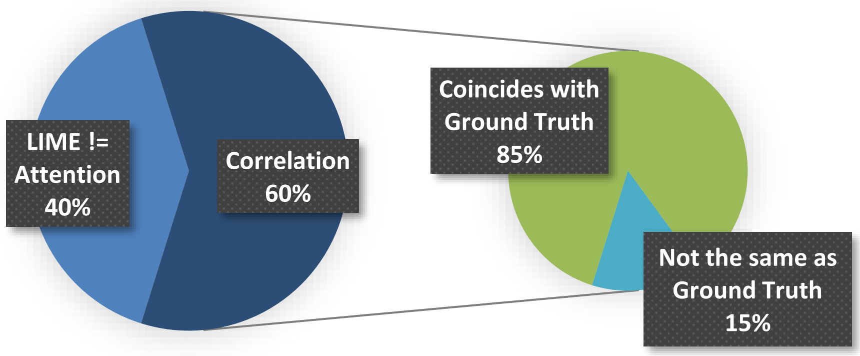


Figure 24 Correlation analysis between LIME and attention heads

If we focus on the sentences where both LIME and Attention suggested the same offset, i.e., those sentences whose output was correlated, Figure 24 also shows that 85% of those sentences predict the expected ("ground truth") value. In contrast, the remaining 15% represent sentences that, although LIME and Attention agree on the output, do not point to the expected value.

Similarly, the fact that there are a significant number of sentences for which LIME and Attention predicted different values indicates that both models indeed work differently. This implies that if there was a way to combine both models, better results could be obtained.

Figure 25 shows that if LIME and Accuracy could be perfectly combined, meaning that an aggregator could be instantiated that would always return the most favorable result, Acc1 could improve from 63% (LIME) or 60% (Attention) to 73%.

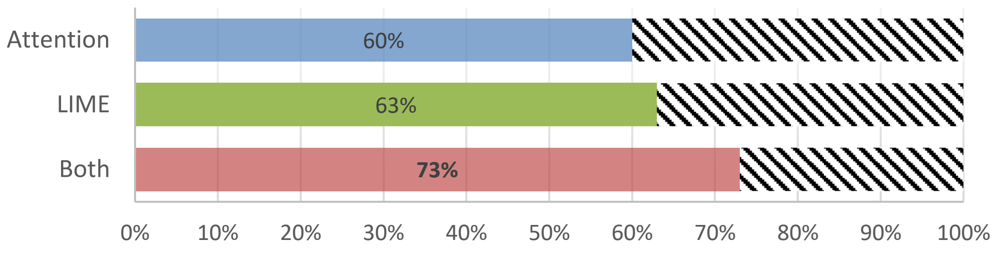


Figure 25 Acc1 scores for LIME, Attention, and for both models combined

4.4 Module 3: Sentence correction

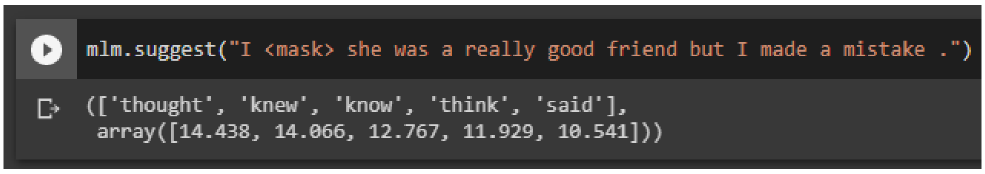
Module 3 is further sub-divided into 2 components: candidate generation and counterfactuals, both are evaluated independently in the following sections.

4.4.1 Candidate Generation

This component is implemented by employing BertForMaskedLM (here referred to as Masked Language Model or “MLM”) to predict candidates to replace a masked token. The model was employed using the standard settings and weights defined by HuggingFace and no fine-tuning took place.

The algorithm is as follows: (1) replace the erroneous token – by means of the index determined in Module 2 – with a special mask-token in the original sentence and (2) feed that sentence to the MLM which, in turn, returns a list of top k most suitable candidates that could be used to replace the masked token. The number *k* here is an arbitrary number: we made the choice to retrieve the 5 topmost tokens. These top k candidate tokens are determined from the entire vocabulary[17] for BERT by the prediction scores of the language modeling head for the masked token. An example of these scores is shown in Figure 26.

To illustrate this, consider the sentence “I *though* she was a really good friend but I made a mistake.” where the word “thought” was misspelled as “though”. By replacing the erroneous token with ‘<mask>’[18] and feeding it to the MLM, we obtain top 5 candidates, along with their prediction scores, which can be seen in Figure 26.



```
mlm.suggest("I <mask> she was a really good friend but I made a mistake .")

([‘thought', 'knew', 'know', 'think', 'said'],
 array([14.438, 14.066, 12.767, 11.929, 10.541]))
```

Figure 26 Example of an output for masked language modeling used in Module 3

This first component is evaluated by means of MRR and Top N Accuracy scores. We do that by evaluating whether the expected correction supplied in the FCE dataset is among the list of candidates generated by the component. Here it is worth noting that we are evaluating the Masked Language Model in its own merit and solely against the expected correction defined in the FCE dataset. In other

[17] *There are 30,552 tokens in total.*

[18] *The mask token <mask> was an arbitrary choice. During processing, it gets substituted with the real mask token as defined by the tokenizer.*

words, we are simply evaluating the degree to which its suggestions (i.e. candidates) match the suggestions of the FCE annotators.

When applied to the single-error development set using the error locations already supplied in the FCE dataset as input, as well as the expected correction provided by the FCE annotators, we obtained an MRR score of 48%. When applied to the test set, the results improved, yielding an MRR score of 52%.

Figure 27 shows the accuracy scores distribution for the development and test datasets. Note that, for 36% of the sentences in the development set, the expected correction was not found among the top 5 candidates returned by the MLM. This improved slightly in the test set, with 34% of the sentences ending without the expected candidate.

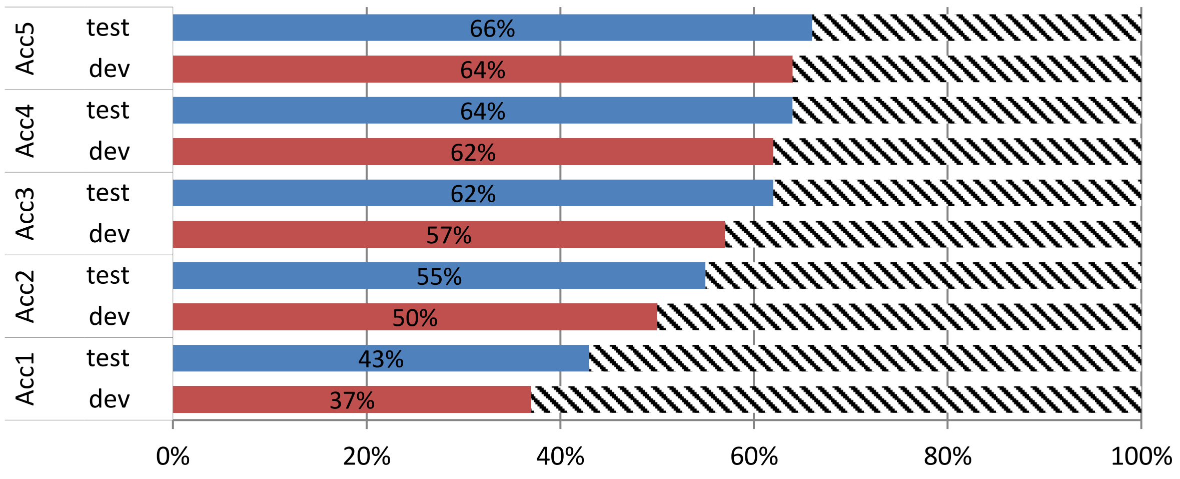


Figure 27 Accuracy scores comparison between development and test sets when evaluated against the ground truth

4.4.2 Counterfactuals

As we have seen already, this component concerns itself with validating the candidates generated by the previous component. Validating here means feeding the corrected sentence back to the model defined in Model 1 (i.e. BERT fine-tuned) and using only the candidates that result in a sentence classified as grammatically by BERT.

The algorithm is as follows: for each candidate generated by the previous component, (1) create different versions of the original erroneous sentence by replacing the erroneous token with the candidates, and (2) validate whether the newly generated "corrected" sentences are actually classified as correct by BERT fine-tuned. If they are, they may be considered as counterfactuals.

The output of this component is a list of one or more correct versions of the original sentence.

The evaluation is done by using the first candidate that results in a (valid) counterfactual according to Module 1's classifier and correcting the original sentence with that candidate. Corrections resulting in a counterfactual that matches the suggestion from the FCE annotators count as a successful hit.

When applied to the single-error development set using the error locations already supplied in the FCE dataset as input, 117 sentences out the original 165 are successfully corrected. This represents an accuracy rate of 71%. It is worth repeating that "successfully corrected" here means that the corrected sentences were validated by the fine-tuned model from Module 1.

When evaluated against the single-error test set, the results improved here as well: out of the original 191 erroneous sentences in the dataset, 143 were corrected, yielding an accuracy rate of 75%.

4.4.3 Discussion

At first glance, the MRR score of 48% (on the single-error development set) for the first component (Candidate Generation) may seem low. However, Figure 27 shows that, while the majority of tested sentences had their expected corrected fulfilled by the top candidate, an almost equal number of sentences have accuracy 0, meaning their expected correction was not found at all among the top 5 candidates. At this point, we decided to do a manual analysis of these results with accuracy 0 to better understand the situation.

We randomly sampled 4 sentences from among those on which to do a manual evaluation in this chapter. The full list of sentences can be found in Appendix D. For easier readability, we have underlined the erroneous word in the original sentences.

Table 9 – Random sample of 4 sentences with accuracy 0

#	original	err_index	target	candidates
26	it 's like a joke , but a bit <u>scarely</u> .	9	scary	['funny', 'silly', 'fun', 'serious', 'sad']
7	i felt how <u>luck</u> i was .	3	lucky	['tired', 'weak', 'alive', 'close', 'happy']
80	it made me so <u>dissapoint</u> and angry .	4	disappointed	['sad', 'hurt', 'confused', 'scared', 'angry']
151	it was closed because of the staff <u>trainning</u> .	7	training	['strike', 'shortage', 'cuts', 'shortages', 'problems']

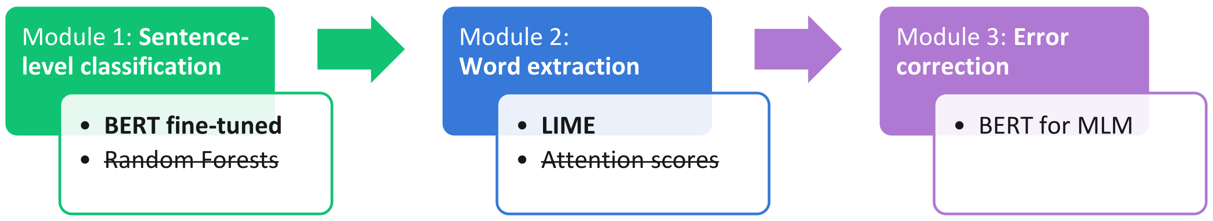
Looking at the sentences in Table 9, we notice that, although the candidates do not match the target, they are, nonetheless, not strictly wrong or incorrect either. In all cases, one could use any of the candidates and still have a correct and coherent sentence. The case of sentence 151 is an interesting case: the candidates are closely related to the negative context of the sentence, which has to do with the fact that "*it* was closed": in this case "staff strike/shortage/cuts/shortages/problems" are all perfectly viable candidates.

It becomes evident then, that it is only when one considers the overall context of a sentence that one can really assess whether one candidate is more suitable than another. When considering only isolated sentences without their broader context – which is the case with the FCE, as well as others – it becomes challenging to assess the candidates. Furthermore, the corrections in the FCE dataset were suggested by an annotator and, as we have seen in chapter 3.5, lack of inter-annotator agreement is an issue that should be taken into account here, as two annotators might choose to correct a same erroneous sentence in different ways.

Finally, sentences 26, 7 and 80 expose a weakness of the Candidate Generation component of Module 3: the original errors are due to the fact that the original words have been misspelled (e.g. "scarely" vs "scary"). In all three cases, the correction would be an adjective and the candidates are as well. However, a spellchecking component, or a combination of MLM and spellchecking, would have had more success. Note that spelling mistakes make up around 21% of the errors across the FCE dataset.

4.5 Entire pipeline

This work proposes using a pipeline with three components, each of which has been evaluated independently. Nevertheless, errors generated from an upper layer propagate further downstream, affecting the performance of the whole pipeline. This section evaluates the effect of each component on the single-error test dataset. Figure 28 shows the models that have shown the best performance for each Module, which is why they have been selected for the final evaluation.



Module 1: Sentence-level classification
BERT fine-tuned
~~Random Forests~~
Module 2: Word extraction
LIME
~~Attention scores~~
Module 3: Error correction
BERT for MLM

Figure 28 Selected models of each Module of the error detection system.

4.5.1 Evaluation results

Figure 29 presents how each Module in the pipeline affects the dataset. The description is as follows:

- We start with 323 single-error sentences in the test dataset, as described in section 4.1.3.
- Out of those, 132 (40%) are mistakenly classified as correct by Module 1. This means that only 191 can be passed to Module 2, as seen in section 4.3.
- The 191 sentences passed to Module 2 are sent to Module 3 for error correction. 48 (25%) of these could not be corrected, whereas for the remaining 143, a correction was provided.
- Out of the 143 corrections, we are only sure about the 38 sentences whose correction matches the expected word. However, the other 105 sentences could still be correct.

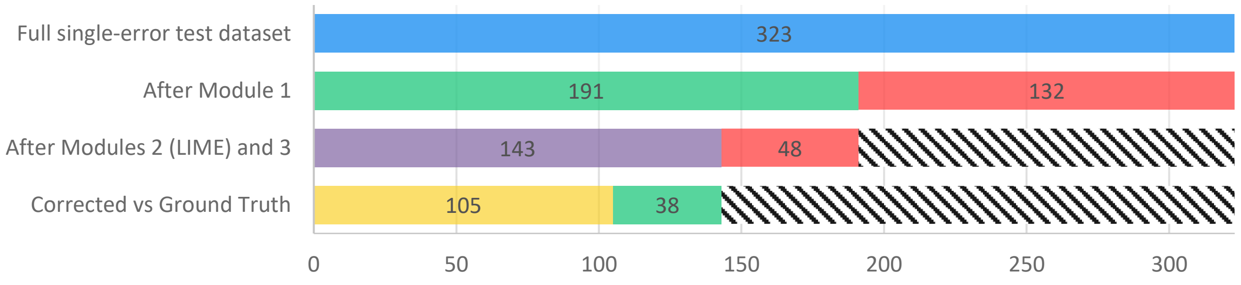


Figure 29 Effect of each Module on the single-error test dataset

A further check is required for those 105 sentences for which a counterfactual was provided, yet the correction did not match the expected word. A professional linguist was consulted for a final annotation to assess them. The complete analysis can be found in Appendix E.

Out of the 105 sentences passed for analysis, 77 were annotated to be grammatically correct, whereas the remaining 28 were still judged to contain an error of some sort. Starting with the 77 correct annotations, it was seen that 59 of those sentences were corrected in the same position as the ground truth suggested, meaning that Module 2 was able to locate the error correctly. The discrepancy on those came from the fact that Module 3 did not suggest the exact same correction, though the error was solved. It could be, though, that the original context was not preserved. Then, regarding the remaining 18 sentences in this block, those were sentences that could have been corrected in more than one way since they turned out to be correct after a correction done in a position different than the one suggested by the ground truth.

A more careful analysis was done on the 28 sentences that the annotator considered erroneous. 8 of those sentences were annotated in the exact same way as the ground truth, which means that our system could not correct the sentence in any way. Another 4 sentences contain annotations that are aligned with the ground truth and point to the word that was corrected by the system, so these are also to be considered to be erroneously corrected. The remaining 16 included different annotations than the ground truth, which points out to the problem of inter-annotation agreement already discussed in this Thesis. These 16 sentences will neither be considered as a hit nor as a miss.

With the results in place, the final evaluation is summarized in Figure 30. To the 38 sentences whose correction matched perfectly the ground truth, we added the 77 sentences annotated as correct. The 28 sentences annotated to be incorrect were split into 12 that are surely wrong, plus 16 sentences that would need extra attention.

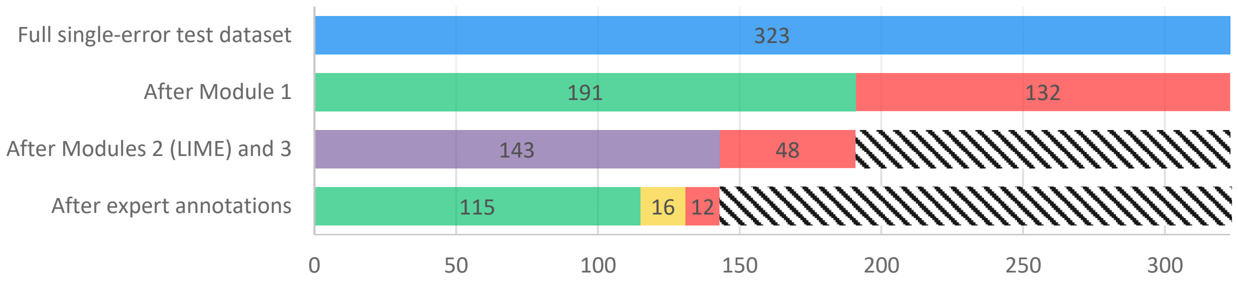


Figure 30 Effect of each Module on the single-error test dataset after results have been annotated

4.5.2 Discussion

As it can be seen, the biggest bottleneck of the system appears to be Module 1, which is discarding (i.e., predicting as correct) 40% of the incoming sentences. This contrasts with the very good results that Module 1 obtained on the whole test dataset (AUC of 81.95%, and accuracy of 80.4%). Both cases differ in the kind of sentences evaluated in each case. The single-error test dataset contains sentences with only one error, and these sentences are more complicated to classify than sentences with multiple errors. Another possible source could be that not all sentences annotated to be incorrect were actually incorrect, although a more detailed analysis would need to be carried out to prove it.

However, when it comes to Modules 2 and 3, which use model explainers, results appear to be very promising. Out of the 191 sentences that passed the filter in Module 1, the system provided a satisfactory correction in at least 115 cases, which yields a correction rate of 60%. If we focus on the sentences for which a counterfactual was found (145), the correction rate goes up to 80%.

In the previous section, we highlighted 8 sentences whose annotation and ground truth coincide. These are sentences 24, 29, 41, 70, 80, 83, 155, and 158. Sentence 70 is an example, which originally was “the rest you already knows .” Our system extracted “already” to be the erroneous word, and suggested the following correction: “the rest you **barely** knows .” Indeed, the pipeline did not capture the wrong token, and the correction did not change the correctness of the sentence.

The 4 sentences pointing to a mistake of the system were 41, 63, 118, and 173. Taking sentence 173 as an example: “I am looking forward to see you soon .” The system detected “forward” to be incorrect and proposed: “I am looking **out** to see you soon .” The annotator now found that both the correction “out” and the original error “see” are present.

Finally, an example of inter-annotator agreement can be found in sentence 67: “it is one of the most important investigation all over the world .” where “investigation” should have been “invention”. Module 2 finds “investigation” to be problematic here and suggests “it is one of the most important **museums** all over the world .” Nevertheless, the sentence is found to be incorrect by the annotator due to “all over”, proposing the correction “it is one of the most important museums **in** the world .”. “all over” was thought to be correct by the original annotator, so we cannot draw any conclusions from these sentences.

5 Conclusions

5.1 Research findings

This work proposes a novel approach to tackle different Grammatical Error tasks using various explainability techniques. A pipeline structure was suggested, after which a grammatically incorrect sentence could be detected and corrected.

The first Module classifies sentences depending on whether they contain a grammatical error. If a sentence is classified as incorrect, Module 2 extracts the word that causes the error. Finally, the third Module uses the original sentence together with the error location to propose a correction.

In chapter 1.3, we defined our research question, which we reproduce below:

How suitable are explainability techniques for solving Grammatical Error tasks?

We have proven that explainability techniques can be used to solve different Grammatical Error tasks. Explainers such as LIME can be used to outline which word in a grammatically incorrect sentence is causing the sentence to be incorrect, hence helping to solve the Grammatical Error Detection task. Similarly, another explainability technique, such as counterfactuals, can be used to suggest a correction to that incorrect sentence, solving the Grammatical Error Correction task.

Furthermore, the following sub goals were defined. After each of them, the most interesting research findings are presented.

- *Develop and train a model capable of detecting if a sentence contains an error.*

After extensive research on the area of Grammatical Error Detection based on Deep Learning, we chose to use BERT as the main model to perform this task as it represents the state-of-the-art in the field. Also, a comparison was run with the Random Forests model as a baseline.

- *Test the models to verify that they can accomplish the task.*

Results showed that BERT outperformed RF for the sentence-level classification task, achieving an AUC score of 83.9 % on the FCE development dataset, whereas RF obtained an AUC score of 72%, more than 10 points worse than BERT. Therefore, it was decided to continue with BERT fine-tuned as the sentence-level classifier. Another evaluation was done on the FCE test set to test the model generalization, obtaining a similar AUC score (81.95 %).

- *Employ model explainability mechanisms to identify what causes the sentence to be classified as grammatically erroneous.*

Two explainability models were considered for the task, namely LIME and BERT attention heads. LIME is the de-facto model-agnostic explainer in the field, whereas the attention heads in BERT fine-tuned can also provide insights into the model decisions.

- *Compare different model explainability mechanisms to determine which is the most suitable for the task.*

Starting with LIME, a process to determine which hyperparameters worked best for the task was carried out. We found them to be bow = False, kernel_width = 25, mask_string = '[MASK]', feature_selection = 'none', and num_samples = 5000. This model achieves 66% accuracy in the single-error development dataset and a very similar 63% accuracy in the single-error test dataset.

Since BERT was already fine-tuned, the process we followed in the attention heads was to select the head and layer combination yielding best results. Also, a pooling scheme needed to be defined to reconstruct words from subtokens, with could be implemented with different methods. We found layer 9, head 9, and pooling mode sum to perform best in the single-error FCE development dataset, with an accuracy of 67%. Results for the single-error test dataset dropped to 60% accuracy.

A correlation analysis was run between the results by LIME and attention heads which showed that both models work differently. It was observed that if the results were combined perfectly, the accuracy score increased to 73%. Nevertheless, this is a theoretical result, and we concluded that the model that performs best is LIME.

- *Prepare a counterfactual-based Module to suggest correct sentences.*

A two-stage approach was implemented to solve this task. Firstly, a candidate generator is invoked to provide words that should work in the context of the erroneous sentence. BERT MLM is used for this purpose. Secondly, the same model developed earlier in this work is used to ensure that the candidates led to correct sentences.

The candidate generator was evaluated against the single-error development and test datasets, obtaining an MRR score of 48% and 52%, respectively. Nevertheless, we believe that the suggestions are better than what the score reflects since it does not reflect the fact that many words can lead to a correct sentence. As proof of that, when the two stages were evaluated together, the accuracy rate achieved by the single-error development and test datasets was 71% and 75%, meaning that three out of four sentences could be corrected.

The whole system was evaluated with all components in place. Module 1 mistakenly classified 40% of the incoming incorrect sentences as correct, impacting the results overall. Only 35% of the incoming sentences were corrected. Modules 2 and 3, which are using model explainability, seemed to perform reasonably good, being able to provide a valid correction for 60% of the sentences that passed Module 1. Also, attending only to those sentences for which a correction proposal was made, 80% of the corrections turned out to be correct.

5.2 Ethical and societal consequences

The main contribution of this study is providing the scientific community with evidence that explainability mechanisms can be applied to GEC tasks, and that attention scores can be suitably used in that manner. A secondary – yet relevant – contribution comes from our in-depth study of LIME's hyperparameters and how they should be configured for best results when applied to text data.

As the word "consequences" may have a negative connotation, Whittlestone et al. (2019) suggest using the word "implications", instead. We acknowledge that it may be impossible for us to foresee every consequence and impact this work may have and that seemingly inoffensive developments have been put to uses they were not intended to. Nevertheless, given the very nature of this research, we do not anticipate negative implications following as a result. On the contrary: positive implications may follow as a result. A system capable of correcting erroneous sentences is something that may lead to very positive implications in several areas, the most obvious of which is education, where ESL students could make use of such systems to improve their knowledge of the language and writing skills. Indeed, given that the models used in this study are able to be extended to other languages, this does not have to be limited only to English.

5.3 Limitations

The most important limitation comes from one of the delimitations defined in Section 1.4. This was a conscious decision that allowed balancing system complexity and performance, as well as constraints on resources and time. That decision meant that we had to make sure that only single-error sentences were considered from the datasets, which greatly limited the size of the validation and test sets. It must be said, however, that this limitation does not affect the classifier fine-tuned for Module 1, which was trained on all available samples from the training dataset.

Furthermore, we limited this study to evaluating BERT and Random Forest as classifiers. The choice to use BERT was due to its virtual ubiquity in NLP implementations nowadays and its overall excellent performance. Random Forest was chosen due its versatility, its ability to deal with high dimensionality, and low bias. Nevertheless, and despite BERT's strengths, around 40% of incoming sentences where discarded (i.e., predicting as correct) during the final evaluation, what is arguably a strong limiting factor for the artifact built as a result of this research.

Regarding explainability mechanisms, LIME was the choice due to its popularity and simplicity. Another alternative would have been SHAP (Shapley Additive explainability). [19]

Inter-annotator agreement is a further limitation, as discussed in Section 3.5. Since there can potentially be many ways of correcting a sentence, the results may be biased by how certain sentence was supposed to be correct.

Finally, we made use only of the FCE dataset. The BEA-2019 Shared Task introduces other datasets that have been annotated with ERRANT, which means that integrating these other datasets along with the FCE would be straightforward.

5.4 Final conclusions

This study evaluated the suitability of model explainability mechanisms when applied to the tasks of GED and GEC. To that end, we set up a pipeline of three modules designed to solve those tasks. In Module 1, we employed two classifiers, namely BERT and Random Forests, and evaluated each on their predictive capabilities for binary sentence classification. Given that BERT achieved a substantial lead over RF (12 percentage points), it was the sole classifier employed in subsequent modules. In Module 2, we compared LIME and attention head scores to extract the location of the error in sentences. LIME was able to achieve better scores (63%) than attention heads (60%). Finally, in Module 3, we took advantage of BERT's MLM capabilities to generate candidate words to correct the grammatical errors located in Module 3. Without any fine-tuning, the MLM classifier was able to achieve an MRR score of 52% on the test set. Using a counterfactual-based analysis, candidates were then evaluated for grammatical correctness with the classifier defined in Module 1, which resulted in 60% of sentences in the test set being successfully corrected. Hence, the research question was answered, and our findings can be summarized in the following way:

- Explainability mechanisms can be successfully employed in GEC tasks.
- Attention heads are able to focus on particular aspects of language and using their scores to locate errors in sentences achieved decent accuracy (i.e. 60%) in the test set. However, the

[19] *https://shap.readthedocs.io/en/latest/index.html*

results vary enough between different datasets to warrant the conclusion that the best layer/head combination for one dataset might not be the same for another. Here we recommend further investigation.

- The correction of sentences may result in the loss of the original context intended by their authors. One way to mitigate this would be to implement a spellchecker to correct misspelled words before employing the MLM component to generate candidate words for correction. This has been noted as a prime line for future work. However, this is a controversial topic as the lack of inter-annotator agreement can attest to. Here we recommend further investigations in the meaning of ground truth in NLP tasks.

5.5 Future Work

As a consequence of doing this work, the following points have been identified as prime targets for lines of future work.

Enhance attention with a norm-based analysis: The recent work by Kobayashi et al. (2020) shows that, by enhancing the self-attention mechanism to be norm-based, the attention heads are able to place more emphasis on more meaningful tokens. If that is indeed the case, this would suggest that the results we were able to obtain by using the attention scores in Module 2 could be improved, making a norm-based analysis a topic of great potential for future work.

Other datasets: This work has been limited to the use of the FCE dataset. Different datasets to try out could be LANG8, or W&I+Locness[20], both of which have also been annotated by ERRANT.

Other encoders: This work has been limited to the use of BERT, though different versions can also be found. Examples of them are mBERT (Devlin et al., 2018), XLM (Lample & Conneau, 2019) and XLM-RoBERTa (Conneau et al., 2019).

Other explainability mechanisms: Future work could include evaluating other algorithms, like SHAP.

Multilingualism: A logical step for this work would be to support languages other than English. XLM-RoBERTa is a very interesting option, due to the fact that it has been trained on 100 different languages and has been shown to outperform both mBERT and XLM.

Aggregator in Module 2: As presented in Chapter 4.3.3, better results could be obtained if various Modules 2 are instantiated in parallel and their outputs are aggregated. An example of such an aggregator could be based on majority voting if three or more different Modules 2 are in place.

Refine Module 2: Chapter 1.4 delimits this study to sentences with only one error, and of type "replace". Do more research to expand Module 2 capabilities to 1) cover errors of the type "remove" and "missing", and 2) be able to detect more than one erroneous word.

Fine-tune Module 3: As we have seen in chapter 4.4.3, the Candidate Generation component in Module 3 could be improved further. The most obvious conclusion from that discussion is that spellchecking could be used to improve the results. Besides that, fine-tuning the Masked Language Model classifier would potentially result in further benefits.

[20] *https://www.cl.cam.ac.uk/research/nl/bea2019st/#data*

References

Ailani, S., Dalvi, A., & Siddavatam, I. (2019). Grammatical Error Correction (GEC): Research Approaches till now. *International Journal of Computer Applications*, *178*(40), 1–3. https://doi.org/10.5120/ijca2019919275

Akbik, A., Blythe, D., & Vollgraf, R. (2018). Contextual String Embeddings for Sequence Labeling. *COLING 2018*, 1638–1649. Retrieved from https://www.aclweb.org/anthology/C18-1139

Alammar, J. (2018). The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). Retrieved from http://jalammar.github.io/illustrated-bert/

Algorithms and artificial intelligence: CNIL's report on the ethical issues. (2018). Retrieved from CNIL website: https://www.cnil.fr/en/algorithms-and-artificial-intelligence-cnils-report-ethical-issues

Alkhatib, M., Monem, A. A., & Shaalan, K. (2020). Deep learning for Arabic error detection and correction. *ACM Transactions on Asian and Low-Resource Language Information Processing*, *19*(5), 1–13. https://doi.org/10.1145/3373266

Alvarez-Melis, D., & Jaakkola, T. S. (2018). On the robustness of interpretability methods. *ArXiv Preprint ArXiv:1806.08049*.

Bahdanau, D., Cho, K., & Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv:1409.0473 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1409.0473

Bell, S., Yannakoudakis, H., & Rei, M. (2019). Context is Key: Grammatical Error Detection with Contextual Word Representations. *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 103–115. https://doi.org/10.18653/v1/W19-4410

Birch, A., Abend, O., Bojar, O., & Haddow, B. (2016). HUME: Human UCCA-based evaluation of machine translation. *ArXiv Preprint ArXiv:1607.00030*.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/bf00058655

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Bryant, C., Felice, M., Andersen, Ø. E., & Briscoe, T. (2019). The BEA-2019 Shared Task on Grammatical Error Correction. *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 52–75. https://doi.org/10.18653/v1/w19-4406

Bryant, C., Felice, M., & Briscoe, E. (2017). *Automatic annotation and evaluation of error types for grammatical error correction*. Association for Computational Linguistics.

Bryant, C., & Ng, H. T. (2015). How far are we from fully automatic high quality grammatical error correction? *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 697–707.

Büchi, M., Fosch-Villaronga, E., Lutz, C., Tamò-Larrieux, A., Velidi, S., & Viljoen, S. (2020). The chilling effects of algorithmic profiling: Mapping the issues. *Computer Law & Security Review*, *36*, 105367.

Carrillo, A., Cantú, L. F., & Noriega, A. (2021). Individual Explanations in Machine Learning Models: A Survey for Practitioners. *ArXiv Preprint ArXiv:2104.04144*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *ArXiv:1406.1078 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1406.1078

Chodorow, M., Dickinson, M., Israel, R., & Tetreault, J. (2012). Problems in evaluating grammatical error detection systems. *Proceedings of COLING 2012*, 611–628.

Choshen, L., & Abend, O. (2018). Inherent Biases in Reference based Evaluation for Grammatical Error Correction and Text Simplification. *ArXiv Preprint ArXiv:1804.11254*.

Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does BERT look at? An analysis of BERT's attention. *ArXiv*. https://doi.org/10.18653/v1/w19-4828

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., … Stoyanov, V.

(2019). Unsupervised cross-lingual representation learning at scale. *ArXiv Preprint ArXiv:1911.02116*.

Dahlmeier, D., Ng, H. T., & Wu, S. M. (2013). Building a large annotated corpus of learner English: The NUS corpus of learner English. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, 22–31.

Deksne, D. (2016). A new phase in the development of a grammar checker for Latvian. *Frontiers in Artificial Intelligence and Applications*, *289*, 147–152. https://doi.org/10.3233/978-1-61499-701-6-147

Denscombe, M. (2014). *The Good Research Guide: For Small-Scale Social Research Projects: for small-scale social research projects*. McGraw-Hill Education (UK).

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1857 LNCS*, 1–15. https://doi.org/10.1007/3-540-45014-9_1

Doshi-Velez, F., & Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. *ArXiv Preprint ArXiv:1702.08608*. Retrieved from http://arxiv.org/abs/1702.08608

Eisenstein, J. (2018). *Natural language processing*. MIT Press, Georgia Tech, USA.

Elio, R., Hoover, J., Nikolaidis, I., Salavatipour, M., Stewart, L., & Wong, K. (2011). *About Computing Science Research Methodology*. Technical report, Alberta University.

Fabisiak, R. (2019). 15 Best Machine Learning Datasets for Free. Retrieved from https://www.blog.duomly.com/15-best-machine-learning-datasets-for-free/

Felice, M., & Briscoe, T. (2015). Towards a standard evaluation method for grammatical error detection and correction. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 578–587.

Futrzynski, R. (2020). Getting meaning from text: self-attention step-by-step video. Retrieved from https://peltarion.com/blog/data-science/self-attention-video

Grammarly. (2019). How We Use AI to Enhance Your Writing | Grammarly Spotlight. Retrieved from https://www.grammarly.com/blog/how-grammarly-uses-ai/

Granger, S. (2003). Error-tagged learner corpora and CALL: A promising synergy. *CALICO Journal*, *20*(3), 465–480. https://doi.org/10.1558/cj.v20i3.465-480

Graves, A., Mohamed, A., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947

Grundkiewicz, R., Bryant, C., & Felice, M. (2020). A Crash Course in Automatic Grammatical Error Correction. *Proceedings of the 28th International Conference on Computational Linguistics: Tutorial Abstracts*, 33–38. https://doi.org/10.18653/v1/2020.coling-tutorials.6

Hasan, K. M. A., Hozaifa, M., & Dutta, S. (2014). Detection of semantic errors from simple Bangla sentences. *2014 17th International Conference on Computer and Information Technology (ICCIT)*, 296–299. IEEE.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844. https://doi.org/10.1109/34.709601

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Jain, S., & Wallace, B. C. (2019). Attention is not explanation. *ArXiv Preprint ArXiv:1902.10186*.

Johannesson, P., & Perjons, E. (2014). An introduction to design science. In *An Introduction to Design Science* (Vol. 9783319106). https://doi.org/10.1007/978-3-319-10632-8

Kaur, J., Garg, K., & Tech, M. (2014). Hybrid Approach for Spell Checker and Grammar Checker for Punjabi. *International Journal of Advanced Research in Computer Science and Software Engineering*, *4*(6), 2277–128. Retrieved from http://ijarcsse.com/Before_August_2017/docs/papers/Volume_4/6_June2014/V4I5-0443.pdf

Knill, K. M., Gales, M. J. F., Manakul, P. P., & Caines, A. P. (2019). Automatic Grammatical Error Detection of Non-native Spoken Learner English. *ICASSP 2019 - 2019 IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), 8127–8131. https://doi.org/10.1109/ICASSP.2019.8683080

Kobayashi, G., Kuribayashi, T., Yokoi, S., & Inui, K. (2020). Attention is Not Only a Weight: Analyzing Transformers with Vector Norms. *ArXiv*. https://doi.org/10.18653/v1/2020.emnlp-main.574

Korre, K., & Pavlopoulos, J. (2020). ERRANT: Assessing and Improving Grammatical Error Type Classification. *Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, 85–89.

Kovaleva, O., Romanov, A., Rogers, A., & Rumshisky, A. (2019). Revealing the Dark Secrets of BERT. *ArXiv:1908.08593 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1908.08593

Lample, G., & Conneau, A. (2019). Cross-lingual language model pretraining. *ArXiv Preprint ArXiv:1901.07291*.

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. *ArXiv:1508.04025 [Cs]*. Retrieved from http://arxiv.org/abs/1508.04025

Madi, N., & Al-Khalifa, H. S. (2018). Grammatical error checking systems: A review of approaches and emerging directions. *2018 13th International Conference on Digital Information Management, ICDIM 2018*, 142–147. https://doi.org/10.1109/ICDIM.2018.8847020

McCormick, C. (2019). BERT Word Embeddings Tutorial. Retrieved from http://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/

Molnar, C. (2020). *Interpretable machine learning* (1st ed.). Retrieved from https://leanpub.com/interpretable-machine-learning

Napoles, C., Sakaguchi, K., Post, M., & Tetreault, J. (2015). Ground truth for grammatical error correction metrics. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 588–593.

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., & Bryant, C. (2014, December 16). *The CoNLL-2014 Shared Task on Grammatical Error Correction*. 1–14. https://doi.org/10.3115/v1/W14-1701

Ng, H. T., Wu, S. M., Wu, Y., Hadiwinoto, C., & Tetreault, J. (2013, December 16). *The CoNLL-2013 Shared Task on Grammatical Error Correction*. 1–12. Retrieved from https://www.aclweb.org/anthology/W13-3601

Nicholls, D. (2003). The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT. *Proceedings of the Corpus Linguistics 2003 Conference*, *16*, 572–581.

O'Leary, Z. (2017). *The Essential Guide to Doing Your Research Project* (3rd ed.). London: SAGE.

Peters, M. E., Ammar, W., Bhagavatula, C., & Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. *ArXiv:1705.00108 [Cs]*. Retrieved from http://arxiv.org/abs/1705.00108

Rahnama, A. H. A., & Boström, H. (2019). A study of data and label shift in the LIME framework. *ArXiv Preprint ArXiv:1910.14421*.

Rei, M., Crichton, G. K. O., & Pyysalo, S. (2016). Attending to Characters in Neural Sequence Labeling Models. *ArXiv:1611.04361 [Cs]*. Retrieved from http://arxiv.org/abs/1611.04361

Rei, M., & Yannakoudakis, H. (2016). Compositional Sequence Labeling Models for Error Detection in Learner Writing. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1181–1191. https://doi.org/10.18653/v1/P16-1112

Ribeiro, M. T. (2016). LIME - Local Interpretable Model-Agnostic Explanations. Retrieved from https://homes.cs.washington.edu/~marcotcr/blog/lime/

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 1135–1144. https://doi.org/10.1145/2939672.2939778

Schmaltz, A. (2019). Toward Grammatical Error Detection from Sentence Labels: Zero-shot Sequence Labeling with CNNs and Contextualized Embeddings. *CoRR*.

Text Classification with Movie Reviews. (n.d.). Retrieved from https://www.tensorflow.org/hub/tutorials/tf2_text_classification

Vashishth, S., Upadhyay, S., Tomar, G. S., & Faruqui, M. (2019). Attention interpretability across nlp tasks. *ArXiv Preprint ArXiv:1909.11218*.

Wachter, S., Mittelstadt, B., & Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, *31*, 841.

Wang, Q., & Tan, Y. (2020). Grammatical Error Detection with Self Attention by Pairwise Training. *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–7. https://doi.org/10.1109/IJCNN48605.2020.9206715

Whittlestone, J., Nyrup, R., Alexandrova, A., Dihal, K., & Cave, S. (2019). Ethical and societal implications of algorithms, data, and artificial intelligence: a roadmap for research. *London: Nuffield Foundation*.

Wiegreffe, S., & Pinter, Y. (2019). Attention is not not explanation. *ArXiv Preprint ArXiv:1908.04626*.

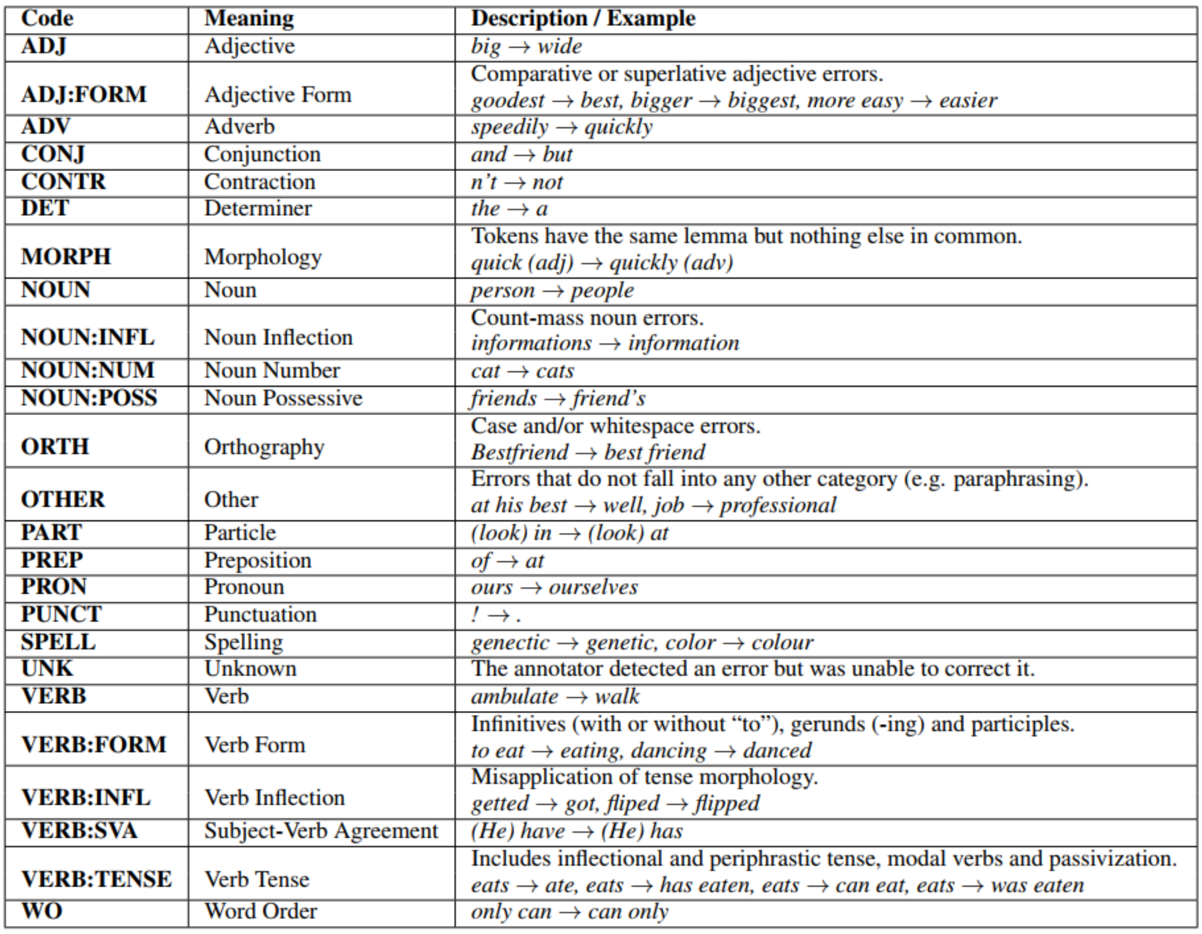
Wikipedia. (n.d.). Algorithmic bias. Retrieved from Wikipedia website: https://en.wikipedia.org/wiki/Algorithmic_bias

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., … Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv Preprint ArXiv:1609.08144*. Retrieved from http://arxiv.org/abs/1609.08144

Yannakoudakis, H., Briscoe, T., & Medlock, B. (2011). A new dataset and method for automatically grading ESOL texts. *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, *1*, 180–189.

Appendix A – ERRANT Error Types

Table 10 – Main error categories in ERRANT (Bryant et. Al, 2017).



Code	**Meaning**	**Description / Example**
ADJ	Adjective	*big → wide*
ADJ:FORM	Adjective Form	Comparative or superlative adjective errors. *goodest → best, bigger → biggest, more easy → easier*
ADV	Adverb	*speedily → quickly*
CONJ	Conjunction	*and → but*
CONTR	Contraction	*n't → not*
DET	Determiner	*the → a*
MORPH	Morphology	Tokens have the same lemma but nothing else in common. *quick (adj) → quickly (adv)*
NOUN	Noun	*person → people*
NOUN:INFL	Noun Inflection	Count-mass noun errors. *informations → information*
NOUN:NUM	Noun Number	*cat → cats*
NOUN:POSS	Noun Possessive	*friends → friend's*
ORTH	Orthography	Case and/or whitespace errors. *Bestfriend → best friend*
OTHER	Other	Errors that do not fall into any other category (e.g. paraphrasing). *at his best → well, job → professional*
PART	Particle	*(look) in → (look) at*
PREP	Preposition	*of → at*
PRON	Pronoun	*ours → ourselves*
PUNCT	Punctuation	*! → .*
SPELL	Spelling	*genectic → genetic, color → colour*
UNK	Unknown	The annotator detected an error but was unable to correct it.
VERB	Verb	*ambulate → walk*
VERB:FORM	Verb Form	Infinitives (with or without "to"), gerunds (-ing) and participles. *to eat → eating, dancing → danced*
VERB:INFL	Verb Inflection	Misapplication of tense morphology. *getted → got, fliped → flipped*
VERB:SVA	Subject-Verb Agreement	*(He) have → (He) has*
VERB:TENSE	Verb Tense	Includes inflectional and periphrastic tense, modal verbs and passivization. *eats → ate, eats → has eaten, eats → can eat, eats → was eaten*
WO	Word Order	*only can → can only*

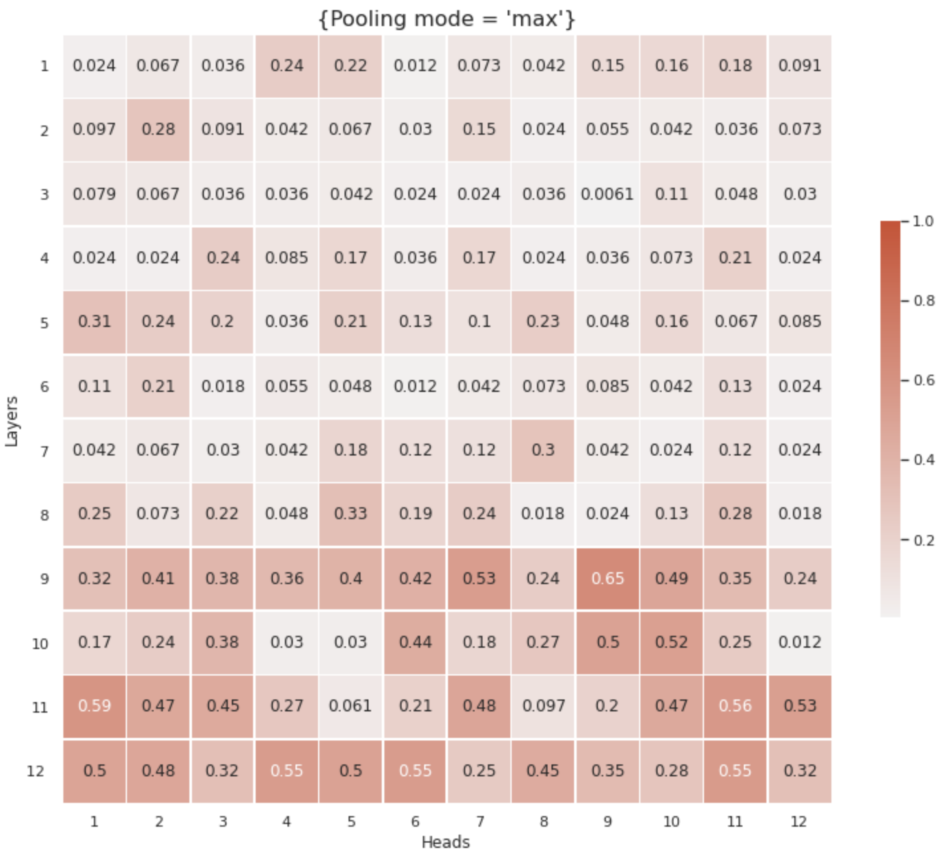
Appendix B – Random Forest configurations

Table 11 – Accuracy and AUC of different RF configurations. Trained on the full train dataset and validated on the development dataset.

Tokenizer	ngram_range	Criterion	Estimators	Accuracy	AUC
TF-IDF	1,1	gini	500	0,76262874	0,704189092
TF-IDF	1,1	gini	600	0,762138303	0,703445044
TF-IDF	1,1	gini	700	0,763119176	0,704933139
TF-IDF	1,1	gini	800	0,763609613	0,70605547
TF-IDF	1,1	gini	900	0,764590486	0,706408716
TF-IDF	1,2	gini	500	0,765571359	0,716975607
TF-IDF	**1,2**	**gini**	**600**	**0,7665522**	**0,718842**
TF-IDF	1,2	gini	700	0,76262874	0,715159303
TF-IDF	1,2	gini	800	0,763609613	0,715512549
TF-IDF	1,2	gini	900	0,763119176	0,714768502
TF-IDF	2,2	gini	500	0,724374693	0,7085701
TF-IDF	2,2	gini	600	0,726826876	0,710398923
TF-IDF	2,2	gini	700	0,725355566	0,709301629
TF-IDF	2,2	gini	800	0,725846003	0,709667394
TF-IDF	2,2	gini	900	0,726336439	0,710033158
CountVectorizer	-	gini	500	0,756743502	0,705095883
CountVectorizer		gini	600	0,758705248	0,705095883
CountVectorizer		gini	700	0,759686121	0,706912186
CountVectorizer		gini	800	0,76115743	0,708766046
CountVectorizer		gini	900	0,758214811	0,706193176
TF-IDF	1,1	entropy	500	0,766552231	0,707115207
TF-IDF	1,1	entropy	600	0,764590486	0,704895583
TF-IDF	1,1	entropy	700	0,764100049	0,704151535
TF-IDF	1,1	entropy	800	0,762138303	0,701931911
TF-IDF	1,1	entropy	900	0,763119176	0,70379829
TF-IDF	1,2	entropy	500	0,768023541	0,715778164
TF-IDF	1,2	entropy	600	0,76949485	0,717632024
TF-IDF	1,2	entropy	700	0,767533104	0,716168966
TF-IDF	1,2	entropy	800	0,768023541	0,71653473
TF-IDF	1,2	entropy	900	0,767533104	0,715790683
TF-IDF	2,2	entropy	500	0,728298185	0,71073965
TF-IDF	2,2	entropy	600	0,727807749	0,710373885

Tokenizer	ngram_range	Criterion	Estimators	Accuracy	AUC
TF-IDF	2,2	entropy	700	0,726336439	0,708898309
TF-IDF	2,2	entropy	800	0,727317312	0,709251554
TF-IDF	2,2	entropy	900	0,727807749	0,709995602
CountVectorizer	-	entropy	500	0,753310446	0,703670381
CountVectorizer		entropy	600	0,7513487	0,70182904
CountVectorizer		entropy	700	0,75282001	0,702926333
CountVectorizer		entropy	800	0,754781756	0,705145957
CountVectorizer		entropy	900	0,755272192	0,706268288

Appendix C – Attention scores results for other pooling modes



Layers \ Heads	1	2	3	4	5	6	7	8	9	10	11	12
1	0.024	0.067	0.036	0.24	0.22	0.012	0.073	0.042	0.15	0.16	0.18	0.091
2	0.097	0.28	0.091	0.042	0.067	0.03	0.15	0.024	0.055	0.042	0.036	0.073
3	0.079	0.067	0.036	0.036	0.042	0.024	0.024	0.036	0.0061	0.11	0.048	0.03
4	0.024	0.024	0.24	0.085	0.17	0.036	0.17	0.024	0.036	0.073	0.21	0.024
5	0.31	0.24	0.2	0.036	0.21	0.13	0.1	0.23	0.048	0.16	0.067	0.085
6	0.11	0.21	0.018	0.055	0.048	0.012	0.042	0.073	0.085	0.042	0.13	0.024
7	0.042	0.067	0.03	0.042	0.18	0.12	0.12	0.3	0.042	0.024	0.12	0.024
8	0.25	0.073	0.22	0.048	0.33	0.19	0.24	0.018	0.024	0.13	0.28	0.018
9	0.32	0.41	0.38	0.36	0.4	0.42	0.53	0.24	0.65	0.49	0.35	0.24
10	0.17	0.24	0.38	0.03	0.03	0.44	0.18	0.27	0.5	0.52	0.25	0.012
11	0.59	0.47	0.45	0.27	0.061	0.21	0.48	0.097	0.2	0.47	0.56	0.53
12	0.5	0.48	0.32	0.55	0.5	0.55	0.25	0.45	0.35	0.28	0.55	0.32

Figure 31 Top 1 Accuracy results for pooling mode "max", for all layer/head combinations, evaluated on the single-error development set.

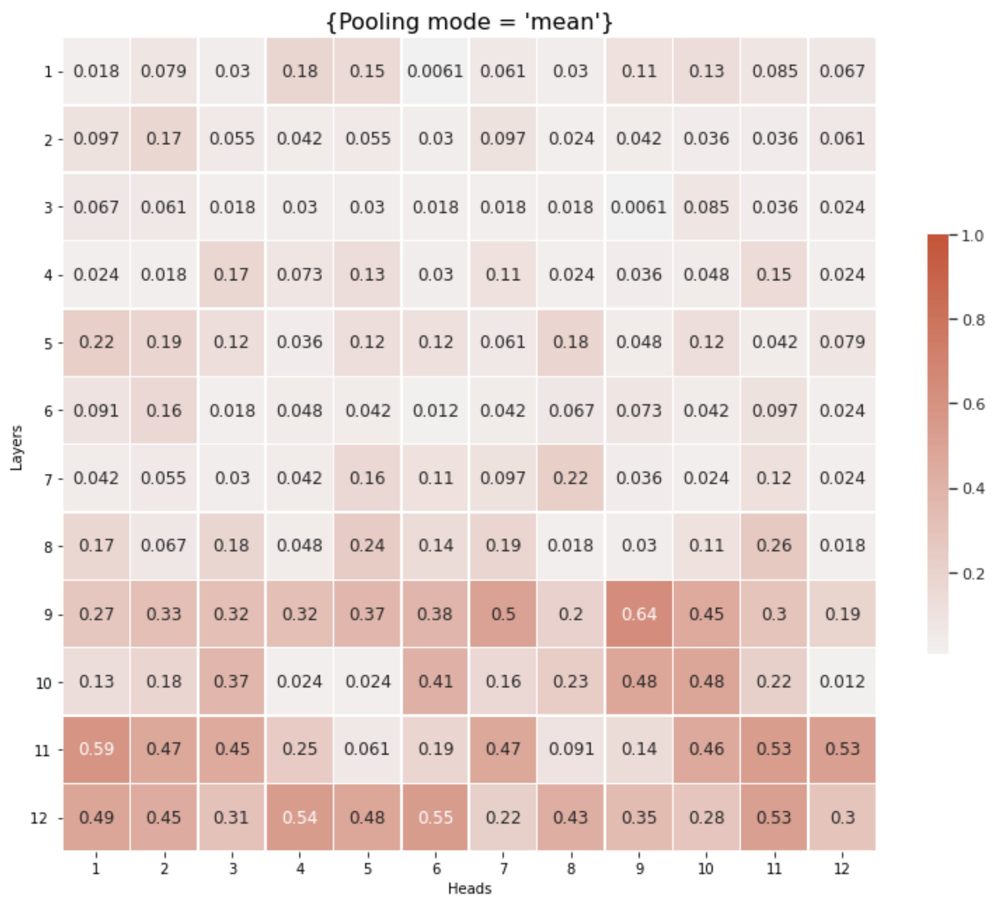


Figure 32 Top 1 Accuracy results for pooling mode "mean", for all layer/head combinations, evaluated on the single-error development set.

Appendix D – Module 3 – List of sentences with accuracy 0

#	original	err_index	target	candidates
6	people all over the world like going to do shopping with friends of families (it 's easier to choose the really needed things for kitchen or bathroom , to get something tasty for dinner , lunch or breakfast) .	13	family	['mine', 'yours', 'ours', 'theirs', 'hers']
7	i felt how luck i was .	3	lucky	['tired', 'weak', 'alive', 'close', 'happy']
9	thanks for your attentions .	3	attention	['help', 'support', 'time', 'concern', 'advice']
10	i was punished , nearly expelt , but pat did n't receive any punishment .	5	expelled	['killed', 'died', 'severely', 'raped', 'certainly']
13	while we were in florence my wallet was stolen , probably by a gypsie .	13	gypsy	['thief', 'man', 'friend', 'criminal', 'stranger']
14	forth , the theatre restaurant was closed .	0	fourth	['subsequently', 'later', 'also', 'eventually', 'however']
19	the latest fashion will be on show and also the leisure and sports wear . really interesting !	2	fashions	['fashion', 'clothes', 'products', 'clothing', 'wear']
21	this is a great opportunity for us to see the latest fashions and famous fashion models who we would like to have authograps from .	22	autographs	['advice', 'lessons', 'fun', 'inspiration', 'information']
22	i noticed that the artists were from only six countries insted that from around the world .	10	instead	['and', ',', 'but', 'besides', 'like']
26	it 's like a joke , but a bit scarely .	9	scary	['funny', 'silly', 'fun', 'serious', 'sad']
27	finally , when i asked about the discounts , an agressive employee refused to answer me .	10	aggressive	['office', 'older', 'old', 'agency', 'elderly']
29	while i 'm stayin there i would like to practise tennis and basketball , as i have been playing both sports all my life , and i have to say that i am very good at both .	3	staying	['not', 'still', 'studying', 'working', 'in']
31	yours faithfuly	1	faithfully	['truly', 'only', 'forever', 'too', 'alone']
32	it was so exaiting ! ! !	3	exciting	['beautiful', 'hot', 'good', 'close', 'loud']
35	so , i told pat that lynne was ugly , fat like a cow , and extremely agressive .	17	aggressive	['ugly', 'dangerous', 'thin', 'beautiful', 'powerful']
41	you must control your behaviour and try not implove so much yourself .	8	implove	['react', 'act', 'change', 'worry', 'worrying']
49	is there a possibility to stay in a log cabin .	3	chance	['reason', 'way', 'place', 'choice', 'need']
51	it seems to be good to change the programme of 14th into the london fashion and leisure show , instead of the science museum and shopping .	11	to	[',', 'at', 'and', 'of', 'annual']
54	i use technology every moment but i did not realise .	4	minute	['day', 'time', 'night', 'year', 'week']
56	yours sincerelly ,	1	sincerely	['truly', 'too', 'alone', 'forever', 'first']
59	we have noticed the environmential damage in recent years .	4	environmental	['greatest', 'most', 'same', 'worst', 'extensive']
62	for example we can buy food , clothes and whatever we need by internet or by telephone .	13	internet	['mail', 'car', 'email', 'truck', 'air']

#	original	err_index	target	candidates
63	i am writing with regard to your advertisment .	7	advertisement	['work', 'request', 'family', 'situation', 'progress']
64	working long hours , doing plenty of activieties , going out , going on holidays .	7	activities	['work', 'shopping', 'things', 'research', 'laundry']
67	according to your advertisment i could have one but in reality there were no discounts at all .	3	advertisement	['advice', 'mother', 'account', 'calculations', 'story']
69	let foreign students do more excises about english to help them pass the exam .	5	exercises	['learning', 'research', 'talking', 'reading', 'information']
71	one of the reasons for my visiting your show was danny brook 's participance .	13	participation	['death', 'absence', 'appearance', 'name', 'father']
72	and that is allways what we want .	3	always	['exactly', 'not', 'precisely', 'just', 'really']
74	last but not least there are some things i would like to know : what kind of wheather do you have in california ?	17	weather	['job', 'family', 'business', 'school', 'friends']
77	hopping to hear from you	0	hoping	['good', 'glad', 'nice', 'great', 'happy']
79	i was very hopefull that i was going to have a good time .	3	hopeful	['excited', 'sure', 'glad', 'happy', 'confident']
80	it made me so dissapoint and angry .	4	disappointed	['sad', 'hurt', 'confused', 'scared', 'angry']
84	defineteley , i want my money back as soon as possible .	0	definitely	['besides', 'plus', 'anyway', 'no', 'still']
89	i had ever never sit down there .	4	sat	['been', 'gone', 'looked', 'come', 'ventured']
90	i had to pay full prise .	5	price	['tuition', 'attention', 'wages', 'costs', 'fees']
91	it all started when one of the organizers asked me to help him at a concert of my favorite band .	18	favourite	['favorite', 'new', 'old', 'own', 'college']
92	first of all because of the big variety in the store .	6	great	['sheer', 'wide', 'large', 'rich', 'limited']
97	many industries throw their contaminated air away , without cleaning it before .	11	first	['up', 'clean', 'away', 'out', 'off']
101	shoping is not always enjoyable .	0	shopping	['it', 'this', 'life', 'sex', 'that']
102	and instead of going to school , why not let school come to you ? we will be learning at home using computers and disc .	24	discs	['technology', 'computers', 'software', 'science', 'stuff']
106	probably it is one of the moments when you want something baddly and you ca n't have it .	11	badly	[',', 'else', 'bad', '.', 'good']
110	best regardes !	1	regards	['friend', 'friends', 'man', 'of', 'picture']
112	i think you should check the information before print it !	8	printing	['sending', 'using', 'publishing', 'reading', 'releasing']
114	i was enthusiatic about receiving your letter . i will give you all the necessary information .	2	enthusiastic	['sorry', 'calling', 'nervous', 'informed', 'worried']
116	nobody likes to be observed constantly or touched by strangers and pointed out in the streets with no right even or just a little privacy .	20	to	[',', 'and', '.', '-', 'with']
118	although all that may change our houses may be still the same .	4	happen	['be', 'remain', 'become', 'haunt', 'fill']
128	at the begining i would like to thank you for your letter - it pleased me much .	2	beginning	['moment', 'end', 'last', 'least', 'earliest']
132	a huge que that lasts half an hour or more and finishes with you completely freaked out .	2	queue	['movie', 'one', 'fight', 'party', 'show']
138	finally , i am considering about what kind of clothes i should wear and how much i need .	4	wondering	['thinking', 'talking', 'unsure', 'deciding', 'confused']

#	original	err_index	target	candidates
139	i would be most grateful if you could give me further advicable information .	11	advicable	['background', 'personal', 'more', 'useful', 'important']
141	for example if you want a new track suit you know that you have to go to a sport shop but if you do n't know what you want you are going to waste your time in the shops .	18	sports	['clothes', 'bike', 'beauty', 'shoe', 'suit']
143	apart from this , i do n't spent my free time in technology .	11	using	['studying', 'on', 'in', 'learning', 'researching']
145	this people will be worried about what they look like .	0	these	['most', 'the', '"', 'some', 'your']
147	everybody deserve to have a private life even if he is a plumber or even if she is a famous singer .	1	deserves	['wants', 'has', 'needs', 'likes', 'seems']
149	i looked at her and she became pale and suddently left the class .	9	suddenly	['she', 'we', 'i', 'quickly', 'then']
150	it was writen that the stars would be danny brook and tina truelove .	2	written	['decided', 'announced', 'expected', 'rumored', 'agreed']
151	it was closed because of the staff trainning .	7	training	['strike', 'shortage', 'cuts', 'shortages', 'problems']
152	first , i had a great expectation to see danny brook plays his role , but he had never shown up that night .	11	playing	['in', 'reprise', 'reprised', 'for', 'resume']
153	moreover , you could face an agressive and maybe even drunk shop assistant or manager .	6	aggressive	['angry', 'alcoholic', 'idiot', 'drunk', 'abusive']
158	i hope you are going to be take into consideration my terrible situation .	12	experience	['situation', 'circumstances', 'choices', 'behavior', 'news']

Appendix E – Final annotations

Description of the task

An extract of the following dataset, containing only the columns “corrected” was sent to the annotator. “label”, “offset”, and “annotation” were provisioned and empty. The task was described as follows:

This dataset has been obtained after passing incorrect sentences through a Deep Learning-based system with the aim of providing corrected versions of the original sentences. The dataset should only contain grammatically correct sentences.

We would like to verify whether all sentences are indeed correct, or whether there are still incorrect sentences among them. This will help us evaluate our system.

The algorithm is as follows. For each sentence,

- *Assess if the sentence is grammatically correct or not. Please, add a "1" in the column "label" if the sentence is annotated to be incorrect, and a "0" if the sentence is annotated to be correct.*
- *If the sentence is classified as incorrect, annotate which word(s) (in case of more than one word, separate them by commas) and the offset where it can be found (separated by dash) in the column "offset". An example is provided below.*
- *If the sentence is classified as incorrect and the word(s) could be annotated, provide a correction in the column "annotation" (in case of more than one word, separate them by commas). Alternatively, if the sentence is too complex to fix, provide a correction in the “annotation” column.*

Example: These is and sentence .
"label": 1
"offset": These-1, and-3
"annotation": This, an

Results

The results are presented in Table 12. Four groups can be observed:

1. The first group only contains column “#”, which represents the sentence ID.
2. “original”, “error_token”, “err_index”, and “word” are the original sentence, as well as which token is erroneous, its position within the sentence, and the expected correction, respectively. This is referred to as the ground truth.
3. “error_loc”, “correction”, and “corrected” represent the indexes by Module 2, the proposed counterfactual by Module 3, and the suggested sentence (expected to be correct).
4. “label”, “offset”, and “annotation” are the columns to be filled by the annotator, as described above.

Table 12 – Dataset to be annotated, consisting of 105 counterfactuals provided by the entire pipeline

#	original	error_ token	err_ index	word	error_ loc	correction	corrected	label	offset	annotation
3	on the top of the tower it has a huge clock on eatch side .	eatch	12	each	12	its	on the top of the tower it has a huge clock on its side .	1	on-1, it-7	The top of the tower has a huge clock on its side.
5	the hystorical museum	hystorical	1	HISTORICAL	1	british	the british museum	0		
8	i heard that a group of america students has just arrived in our town .	america	6	American	6	college	i heard that a group of college students has just arrived in our town .	0		
9	apart from the main hall , there are also lots of other gallaries .	gallaries	12	galleries	12	rooms	apart from the main hall , there are also lots of other rooms .	0		
11	it is based in the city centre and is about half an hour by underground from the conference .	by	2	located	13	walk	it is based in the city centre and is about half an hour walk underground from the conference .	1	half-11, an-12, hour-13	it is based in the city centre and is about a half - hour walk underground from the conference .
13	but if i had to write a letter and to send it by post , the money will arrive two or three days after .	had	17	would	3	have	but if i have to write a letter and to send it by post , the money will arrive two or three days after .	1	after-24	later
14	it is not an easy desicion how to react now .	desicion	5	decision	5	question	it is not an easy question how to react now .	0		
16	the bus will pick you up right at your hotel entery .	entery	10	entrance	10	room	the bus will pick you up right at your hotel room .	0		
17	firstly , there is the acquerium in the building .	acquerium	5	aquarium	5	elevator	firstly , there is the elevator in the building .	0		
18	i 'm looking forward to seeing you on the conference .	conference	7	at	9	road	i 'm looking forward to seeing you on the road .	0		
19	on another situation , your son is ill , he needs help urgently .	situation	0	In	2	note	on another note , your son is ill , he needs help urgently .	0		

#	original	error_ token	err_ index	word	error_ loc	correction	corrected	label	offset	annotation
24	thirdly , the end - of - conference party would be a small disco , with a variety of drinks and food provided .	conference	9	will	7	season	thirdly , the end - of - season party would be a small disco , with a variety of drinks and food provided .	1	would-10	will
29	your hotel is called palace hotel and it is placed in the city centre .	palace	9	located	4	the	your hotel is called the hotel and it is placed in the city centre .	1	placed-10	located
30	it must be a very interesting visit for the students who has never experienced an oriental atmosphere .	students	11	have	9	one	it must be a very interesting visit for the one who has never experienced an oriental atmosphere .	1	the-9 , one-10	someone
31	i hope that our friend richard brown does n't have any serious willness .	willness	12	illnesses	12	problems	i hope that our friend richard brown does n't have any serious problems .	0		
33	ceartainly it would be just like mine .	ceartainly	0	Certainly	0	and	and it would be just like mine .	0		
34	she win £ 500 ! ! !	she	1	won	0	to	to win £ 500 ! ! !	0		
35	have you ever imaged how this world would be if there was no electric light ?	imaged	3	imagined	3	wondered	have you ever wondered how this world would be if there was no electric light ?	1	electric-14, light-15	electricity
37	i prefer being informed by television than by newspapers , it 's more relaxing .	,	9	;	9	.	i prefer being informed by television than by newspapers . it 's more relaxing .	1	by-5, by-8	i prefer being informed through television rather than through newspapers . it 's more relaxing .
38	there , i saw the exibitions and admired the building itself .	exibitions	5	exhibitions	5	building	there , i saw the building and admired the building itself .	0		
39	i love baking cakes , cockies and breads .	cockies	5	cookies	5	cakes	i love baking cakes , cakes and breads .	1	breads-8	bread
40	i will be pleasure to help you .	pleasure	0	It	3	there	i will be there to help you .	0		
41	at the end - of - conference party it 'll be something to eat and to drink .	end	8	there	2	out	at the out - of - conference party it 'll be something to eat and to drink .	1	it-9	there

#	original	error_ token	err_ index	word	error_ loc	correction	corrected	label	offset	annotation
42	however , the invention of the airplane has brought us some facilities .	some	11	facilities	10	new	however , the invention of the airplane has brought us new facilities .	0		
44	nothing will be spacial .	spacial	3	special	3	done	nothing will be done .	0		
45	i hope my suggestion and idea will help you to organise the meeting .	idea	5	ideas	5	advice	i hope my suggestion and advice will help you to organise the meeting .	0		
46	location : the castle is in the city centre nearby an enormous church .	nearby	9	near	9	with	location : the castle is in the city centre with an enormous church .	0		
49	at the end of the conference , all students and leader will have a drink .	leader	10	leaders	10	faculty	at the end of the conference , all students and faculty will have a drink .	0		
50	what would you have done in that moment without a mobile phone ?	moment	5	at	7	situation	what would you have done in that situation without a mobile phone ?	0		
52	reasons of recommendation .	reasons	1	FOR	0	letter	letter of recommendation .	0		
54	ladies should wear an evening dress and gentlman a dark suit .	gentlman	7	gentlemen	7	not	ladies should wear an evening dress and not a dark suit .	0		
55	thank you for your lettre .	lettre	4	letter	4	support	thank you for your support .	0		
57	there will be a bus taking students who do not need to stay at the party until the end .	taking	10	want	5	for	there will be a bus for students who do not need to stay at the party until the end .	0		
58	yours sincerelly	sincerelly	1	sincerely	1	truly	yours truly	0		
61	in the church are many expensive draws .	church	6	drawings	2	center	in the center are many expensive draws .	1		in the center there are many expensive draws .
62	do n't miss the chance to visit the hystorical museum , located in the heart of the capital city .	hystorical	8	historical	8	national	do n't miss the chance to visit the national museum , located in the heart of the capital city .	0		
63	i look forward to see you at the conference .	forward	4	seeing	2	happy	i look happy to see you at the conference .	1	happy-3, see-5	forward, seeing

#	original	error_ token	err_ index	word	error_ loc	correction	corrected	label	offset	annotation
65	it was a very wonderful party and they have never forgot that day .	they	10	forgotten	7	i	it was a very wonderful party and i have never forgot that day .	1	very-4, have-9, forgot-11	it was a wonderful party and i will never forget that day .
66	yours sincerly ,	sincerly	1	sincerely	1	truly	yours truly ,	0		
67	it is one of the most important investigation all over the world .	investigation	7	inventions	7	museums	it is one of the most important museums all over the world .	1	all-9, over-10	in
68	nowadays , our lives is becoming better and better .	lives	4	are	3	world	nowadays , our world is becoming better and better .	0		
70	the rest you already knows .	already	4	know	3	barely	the rest you barely knows .	1	knows-5	know
71	the momerial hall is surrounded by large playgrounds .	momerial	1	Memorial	1	town	the town hall is surrounded by large playgrounds .	0		
72	there is also a gift shop in there , where your boys and girls can buy some soveniers for their families and friends .	soveniers	17	souvenirs	17	gifts	there is also a gift shop in there , where your boys and girls can buy some gifts for their families and friends .	0		
74	i wait you .	wait	1	wait	1	love	i love you .	0		
79	the end - of - conference party starts just one our after the end of the conference .	our	10	hour	10	week	the end - of - conference party starts just one week after the end of the conference .	0		
80	there is a special monument for the last queen of chosun who was killed by the japanese army .	queen	5	to	8	prince	there is a special monument for the last prince of chosun who was killed by the japanese army .	1	for-6	to
82	you will than have an hour to check in and unpack all your luggage .	than	2	then	2	only	you will only have an hour to check in and unpack all your luggage .	0		
83	if you are free i will show you arround our college .	arround	8	around	8	to	if you are free i will show you to our college .	1	to-9	around
87	the musseum is situated in front of the sea in the centre of acapulco .	musseum	1	Museum	1	stadium	the stadium is situated in front of the sea in the centre of acapulco .	0		
91	you can find any information from it what you want .	what	7	that	7	if	you can find any information from it if you want .	1	from-6	in

#	original	error_ token	err_ index	word	error_ loc	correction	corrected	label	offset	annotation
92	after that i am sure you will be happy to find a few restaurands located at the new castle .	restaurands	13	restaurants	13	rooms	after that i am sure you will be happy to find a few rooms located at the new castle .	0		
94	she appears occationaly and opens every window in the castle .	occationaly	2	occasionally	2	constantly	she appears constantly and opens every window in the castle .	0		
95	whising i 've been very helpful to you .	whising	0	Hoping	0	"	" i 've been very helpful to you .	0		
97	i look forward to meeting you on the conference .	conference	6	at	8	way	i look forward to meeting you on the way .	0		
98	i am looking forward to seeing you and hop you have a safe flight over to england .	hop	8	hope	8	letting	i am looking forward to seeing you and letting you have a safe flight over to england .	1	over-15	i am looking forward to seeing you and letting you have a safe flight to england .
99	everybody an enjoy themselves .	an	1	will	1	should	everybody should enjoy themselves .	0		
101	you can admire every kind of equipments you need to do any particular sport .	equipments	6	equipment	6	skill	you can admire every kind of skill you need to do any particular sport .	0		
103	one is the picasso museum and the other is an important gallery where you can enjoy seeing differents kinds of paintings .	differents	17	different	17	all	one is the picasso museum and the other is an important gallery where you can enjoy seeing all kinds of paintings .	0		
108	i appreciate if i can offer you my help .	appreciate	1	appreciate	1	wonder	i wonder if i can offer you my help .	0		
109	subject : arboriginal art museum .	arboriginal	2	Aboriginal	2	seattle	subject : seattle art museum .	0		
111	i first bought a computer as i was in university .	university	5	when	9	college	i first bought a computer as i was in college .	0		
112	the etruscan museum is one of the most attractive building in my town .	building	9	buildings	9	places	the etruscan museum is one of the most attractive places in my town .	0		
114	there is a very big labrary .	labrary	5	library	5	problem	there is a very big problem .	0		

#	original	error_ token	err_ index	word	error_ loc	correction	corrected	label	offset	annotation
116	firstly , let me explain the way from the palace hotel , where your group will be staying , to the conference .	explain	6	route	4	lead	firstly , let me lead the way from the palace hotel , where your group will be staying , to the conference .	0		
117	we are now still friend .	friend	4	friends	4	alive	we are now still alive .	0		
118	i look forward to see you and your group soon .	forward	4	seeing	2	happy	i look happy to see you and your group soon .	1	happy-3, see-5	forward, seeing
121	personally , i do think televisions is the most important invention .	televisions	6	are	5	it	personally , i do think it is the most important invention .	0		
128	our college is not much far away .	much	4	very	4	too	our college is not too far away .	0		
130	best regardes	regardes	1	regards	1	book	best book	0		
131	then we can have a rest in the caffee and talk before your flight home .	caffee	8	cafe	8	car	then we can have a rest in the car and talk before your flight home .	1	a-5	some
132	before going inside , you have to pas through the huge garden .	pas	7	pass	7	walk	before going inside , you have to walk through the huge garden .	0		
134	and some old toilets might look awlful .	awlful	6	awful	6	old	and some old toilets might look old .	0		
136	it is a different expirience .	expirience	4	experience	4	story	it is a different story .	0		
137	if you go down a few steps , you will discover the ceilar .	ceilar	12	cellar	12	truth	if you go down a few steps , you will discover the truth .	0		
141	i am sorry to hear that richard brown is n't well and hope he 'll get better soon . i am more than happy to give you the nessessary information .	nessessary	28	necessary	28	full	i am sorry to hear that richard brown is n't well and hope he 'll get better soon . i am more than happy to give you the full information .	0		
145	on your last day you have an afternoon free .	free	6	the	8	off	on your last day you have an afternoon off .	0		
146	is n't that contradictorious ?	Contradict-torious	3	ironic	3	right	is n't that right ?	0		

#	original	error_ token	err_ index	word	error_ loc	correction	corrected	label	offset	annotation
154	there are lots of thing there you will enjoy watching or buying for friends .	thing	4	things	4	movies	there are lots of movies there you will enjoy watching or buying for friends .	0		
155	they must get on the bus from the city centre and get off after five stations at palace station .	bus	15	stops	5	train	they must get on the train from the city centre and get off after five stations at palace station .	1	stations-16	stops
156	they can also talk with british students to communiate study experience .	communiate	8	communicate	8	gain	they can also talk with british students to gain study experience .	0		
157	i am writting to give you the information that you asked me for .	writting	2	writing	2	going	i am going to give you the information that you asked me for .	0		
158	you can see the evolution of sports and understand why changes gradually have been done to improve them .	gradually	14	made	11	should	you can see the evolution of sports and understand why changes should have been done to improve them .	1	done-15	made
161	due to the mobile phone .	due	5	,	0	go	go to the mobile phone .	0		
162	todays nobody could imagine such a situation .	todays	0	Today	0	but	but nobody could imagine such a situation .	0		
163	let us try to appreciate thier importance again .	thier	5	their	5	its	let us try to appreciate its importance again .	0		
164	the place i would like to recommend is the seventienth - century royal palace .	seventienth	9	seventeenth	9	sixteenth	the place i would like to recommend is the sixteenth - century royal palace .	0		
166	i hope you will enjoy the visit and i wish you a nice holliday in yverdon .	holliday	13	holiday	13	home	i hope you will enjoy the visit and i wish you a nice home in yverdon .	0		
167	the hotel that group has been booked into is the palace hotel (in the victoria road) .	group	13	on	3	he	the hotel that he has been booked into is the palace hotel (in the victoria road) .	0		
169	that 's the nature of human being .	being	6	beings	6	nature	that 's the nature of human nature .	1	humans-6, nature-7	that's the nature of humans

#	original	error_ token	err_ index	word	error_ loc	correction	corrected	label	offset	annotation
172	i just had no possibility to meet them after school , because i had no vehicle to get there .	possibility	4	way	4	desire	i just had no desire to meet them after school , because i had no vehicle to get there .	0		
173	i am looking forward to see you soon .	forward	5	seeing	3	out	i am looking out to see you soon .	1	out-4, see-6	forward, seeing
174	cars have definitely changed all our lives , and i am not refering only in a positive way , but in a negative way as well .	refering	12	referring	12	thinking	cars have definitely changed all our lives , and i am not thinking only in a positive way , but in a negative way as well .	0		
175	2) secondly , they can enjoy visiting museums , and afterwards they can take something to eat or to drink in the " pizzerias ' and restaurants near these famous places .	'	14	have	25	"	2) secondly , they can enjoy visiting museums , and afterwards they can take something to eat or to drink in the " pizzerias " and restaurants near these famous places .	0		
176	this scaring dark is still affecting my life now .	scaring	1	scary	1	constant	this constant dark is still affecting my life now .	1	dark-3	darkness
177	after this invention travel around the world is so much easier .	invention	0	With	2	,	after this , travel around the world is so much easier .	1	travel-4	traveling
179	we will prepare traditional english meals and some students from foreign country will prepare their ethnic foods .	foreign	11	countries	10	the	we will prepare traditional english meals and some students from the country will prepare their ethnic foods .	1	ethnic-16, foods-17	national, dishes
180	of course it is a pity that mr. brown has been ill , but we are happy that you are comeing now .	comeing	20	coming	20	here	of course it is a pity that mr. brown has been ill , but we are happy that you are here now .	0		
181	you do not have to wear spezial clothes , just wear what you always wear .	spezial	6	special	6	any	you do not have to wear any clothes , just wear what you always wear .	0		

#	original	error_ token	err_ index	word	error_ loc	correction	corrected	label	offset	annotation
182	because there are a lot of beatiful trees and you can see the sea .	beatiful	6	beautiful	6	palm	because there are a lot of palm trees and you can see the sea .	0		
184	if you need futher information do not hesitate to contact us .	futher	3	further	3	any	if you need any information do not hesitate to contact us .	0		
186	" televisions is just an amazing invention . " my good friend said that . i agree with her .	televisions	1	Television	1	this	" this is just an amazing invention . " my good friend said that . i agree with her .	0		
187	the group has been booked into the harris hotel .	has	0	The	2	had	the group had been booked into the harris hotel .	0		
188	i am writing in reply to your letter about the internation student conference .	internation	10	international	10	upcoming	i am writing in reply to your letter about the upcoming student conference .	0		
190	after that our college has organised a barbequiou night , with traditional local music that you must not miss .	barbequiou	7	barbecue	7	music	after that our college has organised a music night , with traditional local music that you must not miss .	0		

Appendix E – Enrique’s reflection

I’m about to finalize my studies in the context of the Master’s Programme in Decision analysis and Data Science. So, in that sense, a Thesis written in the field of Data Science, and more specifically, in such a trendy topic like NLP, was really exciting.

I think that the Thesis work is really the icing on the cake of the studies. It should show how we have put all the knowledge that we acquired during the Master, and how we can relate different courses altogether. At the same time, it is truly challenging, as you usually find yourself out of the comfort zone, yet with a big toolbox available and lot of enthusiasm.

For this work, I think that the most relevant courses have been Data Mining I and II, Logic, and Programming for Data Science, for the technical part. For the methodological aspect, courses like and Scientific Communication and Research Methodology, and Research Methodology for Computer and Systems Sciences were very useful. When doing research of any kind, not only results are important: one must also consider how to present them so that others can understand. In this case, courses about methodology and having a template available have been of great help to organize our own work, sometimes a bit chaotic.

Someone told me once that research is very similar to mining. One certainly needs to work very hard and try out many different things until an idea worth pursuing is found. When we started the Thesis, we just knew that it would verse about GEC and how we could make it better or solve it differently. So, it required a lot of brainstorming, idea polishing, and trial and error. In this regard, I truly appreciate all the effort and commitment shown by John, our supervisor. He did his absolute best to help us in everything he could.

We tried to create a plan at the beginning of the Thesis work, using the required milestones and the critical deadlines. However, things did not always turn up in the way we wanted. At the beginning there were a lot of possibilities and tangent tracks, all of them looking very promising, but at the end we reached cruising speed and finished the Thesis just in time.

Could we have done things differently? Indeed yes, and most likely, the whole process would have been easier. But then our learning wouldn’t have been the same. We learned so much during the process, and not only about the technical aspects involved in NLP and GED. We also learned about how actual research is carried out and the actual challenges that it poses to researchers. This is one of my biggest takeaways from this work.

As already mentioned, I consider the Thesis to be proof of the fact that we learned how to solve Data Science projects. Hence, we should now be able to take on real-world challenges and tackle them independently. I personally consider that Data Science, Machine Learning, and Artificial Intelligence are here to stay, and it is great to be part of it.

Overall, I feel very satisfied with how the Thesis went and the results we obtained. We must not forget that, only 6 months ago, we were only a couple of enthusiasts with very little knowledge about NLP and everything behind it. Now, I think we have been able to show that model explainers can be used to solve certain grammatical error tasks. This is something that, to the best of our knowledge, nobody tried before. Sure, there are things to improve in this field, yet the results seem very promising.

Appendix F – Ronaldo’s reflection

My thesis partner and I have worked closely for the past six months, successfully completing our study. We set out to build a system in an area with which we had no prior experience (namely, natural language processing), and applying techniques and skills in which we had only theoretical experience. What followed were 6 months of hard work and a considerable amount of research, scouring the scientific literature, technical blogs, user guides, etc.; as well as coding, conducting experiments in machine learning, collecting and analyzing data; at the end of which we were able to put together a system that actually works, as well as this present document. I do not believe I speak only for myself when I say that we have learned much and gained many valuable skills in the process.

We started off by making a rough draft of what we wanted to achieve, week by week, starting in mid-January and ending in mid-May. We took turns adding text to the thesis document and conducting experiments, working towards building a working system. However, due to our lack of experience with the topic (i.e. NLP), things quickly got out of control. We found ourselves having to research several topics to which we had not had prior exposure, causing us to miss some of our internal deadlines. At one point, we found ourselves facing a technical issue with LIME that, at the time, caused me to become very concerned with the future of our thesis. Fortunately, we were able to work our way through those challenges, after which we were able to progress much faster. Additionally, we spent a lot of time coming up with toy datasets designed to evaluate our classifiers. If I could have done things differently, I would switched some of that time in searching for existing datasets to use in that phase of our study. Or, if not that, at least some tools that we could have used to automate that process, which would have saved us both time and effort.

Though all courses were important and relevant, to me, the individual courses that had the most bearing on this thesis work were the FMVEK (Scientific Communication and Research Methodology) and both DAMI (Data Mining) classes. FMVEK taught me the necessary skills to carry out all the necessary steps for proper research and was an invaluable class. Seeing that I am a student in the Decision Analysis and Data Science programme in general, and specifically of the Data Science track, both DAMI classes were the actual practical core of the entire program, teaching the necessary theory and practice of machine learning algorithms.

The thesis encompasses all the topics studied along with the course of the Master’s program and enabled me to put that knowledge into use, in the form of an actual practical project. I find it very inspiring that I went from knowing practically nothing in the field of AI and machine learning to finishing a project applying them.
I currently work in IT, and more specifically, business intelligence. Though I have yet needed to apply the knowledge I obtained in the program and the thesis so far, I am now more keenly aware of areas of improvement that might benefit from that. Additionally, data science and machine learning are definitely here to stay, so it is important to be qualified in those areas. Technological progress always brings both opportunities and challenges. Some professions will fall by the wayside, while new ones will arise. I believe that the skills I obtained in the course of doing this thesis gives me the tools necessary to position myself so that I could make the best of the coming changes.

Regarding the thesis work and results, to be honest, due to my naiveté, I started off with ambitions for this thesis that I now realize to be unrealistic. I think this is something that probably happened to other students as well: when one is introduced to a new technological with all the potential that data science and machine learning have, it is only natural to think “Wow! I can do anything with this!” However, it is only when one actually starts to put it into practice that one realizes that things are not that simple. This happened to me. I would have liked to develop a GEC system capable of detecting and correcting **all** errors in sentences. Though I am quite satisfied with the results we were able to secure at the end, I now realize that a couple of students with no prior experience in the field, were not going to be able to beat Grammarly in just 6 months…

Not yet, anyway!