UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

Model robustness under data distribution shifts: analysing and predicting the impact of text perturbations on NLP models.

End of Degree Project

Bachelor's Degree in Data Science

AUTHOR: Romero Alvarado, Daniel

Tutor: Hernández Orallo, José

Cotutor: Martínez Plumed, Fernando

ACADEMIC YEAR: 2022/2023

Resumen

Los grandes modelos de lenguaje natural suelen ser entrenados con datasets pretratados y limpiados de impurezas como faltas de ortografía, contracciones, etc. Por lo tanto, existe una diferencia entre los datos de entrenamiento de estos modelos y los datos que se encuentra en entornos de despliegue. En este trabajo se evalúa la robustez de cuatro modelos de lenguaje en cinco tareas de lenguaje natural frente a entradas perturbadas. Para ello, se analizan tres tipos de perturbaciones: a nivel de carácter, a nivel de palabra, y otros tipos. Los conjuntos de datos son perturbados y sus predicciones se comparan con las predicciones en los conjuntos de datos sin alterar. Los resultados muestran que los modelos son sensibles a las entradas perturbadas, con algunos modelos siendo más sensibles qué otros dependiendo de la tarea y del tipo de perturbación. En concreto, el modelo XLNet es el más robusto en general, y la tarea más sensible es la de coherencia gramatical.

Palabras clave: Procesamiento del Lenguaje Natural, perturbaciones de texto, robustez, transformers, inferencia de lenguaje natural, análisis de emociones, lenguaje ofensivo y discurso de odio, similitud semántica, aceptabilidad lingüística

Abstract

Large language models are usually trained using curated datasets, which lack impurities such as typographic errors, contractions, etc. Therefore, there is a gap between the training data of these models and the data they encounter in deployment situations. This work evaluates the robustness of four models in five different Natural Language Processing tasks against perturbed inputs. For that purpose, three perturbations type are analysed: character level perturbations, word level perturbations, and other types of perturbations. Datasets are perturbed and their predictions are compared against those of the unaltered datasets. Results show that models are sensitive to perturbed inputs, with some models being more sensitive that others depending on the task and the perturbation type. Precisely, the XLNet model is in general the most robust, and the most sensitive task is grammatical coherence.

Keywords : Natural Language Processing, text perturbation, robustness, transformers, natural language inference, sentiment analysis, hate speech and offensive language, semantic similarity, linguistic acceptability

Table of contents

1. Introduction 6
1.1. Motivation 7
1.2. Objectives 8
1.3. Expected impact 9
1.4. Ethical and legal implications 10
2. State of the art and background 11
3. Methodology 16
3.1. Models 16
3.2. NLP tasks 20
3.3. Perturbations 24
3.3.1. Character-level perturbations: 24
3.3.2. Word-level perturbations 25
3.3.3. Other perturbations 25
3.4. The finetuning process: hyperparameter optimisation 27
3.5. The evaluation process: perturbing and predicting sentences 28
3.6. Hardware, reproducibility and reporting results 30
4. Results 31
4.1. Finetuning results: finding the best hyperparameters 31
4.2. Evaluation results 32
4.2.1. Some clarifying examples 32
4.2.2. Detailed analysis 35
4.2.3. Aggregated analysis 49
5. Conclusions 52
6. Legacy and relationship with studies 54
7. Future work and improvements 56
8. Acknowledgments 57
9. Bibliography 58

List of tables

Table 1. Summary of the characteristics of the four models: DistilBERT, ELECTRA, XLnet and Funnel Transformer 20
Table 2. Examples for the Stanford Sentiment Treebank with their corresponding label 21
Table 3. Examples for the Corpus of Linguistic Acceptability with their corresponding label 21
Table 4. Examples of the Paraphrase Adversaries from Word Scrambling dataset with their corresponding label 22
Table 5. Examples of the Multi-Genre Natural Language Inference Corpus dataset with their corresponding label 22
Table 6. Examples of the Multi-Genre Natural Language Inference Corpus dataset with their corresponding label 23
Table 7. Summary of the five datasets: CoLA, HSOL; MNLI, SST2 and PAWS 23
Table 8. Dummary of the perturbations with a short example 26
Table 9. Hyperparameters employed in the finetuning phase 28
Table 10. Example of the finetuning scores for DistilBERT in GC 31
Table 11. Best combination of hyperparemeters 31
Table 12. Cohen's kappa scores for character level perturbations in GC 35
Table 13. Cohen's kappa scores for word level perturbations in GC 36
Table 14. Cohen's kappa scores for other perturbations in GC 38
Table 15. Cohen's kappa scores for character level perturbations in HSOL 39
Table 16. Cohen's kappa scores for word level perturbations in HSOL 40
Table 17. Cohen's kappa scores for other perturbations in HSOL 41
Table 18. Cohen's kappa scores for character level perturbations in NLI 41
Table 19. Cohen's kappa scores for word level perturbations in NLI 42
Table 20. Cohen's kappa scores for other perturbations in NLI 43
Table 21. Cohen's kappa scores for character level perturbations in SA 43
Table 22. Cohen's kappa scores for word level perturbations in SA 45
Table 23. Cohen's kappa scores for other perturbations in SA 45
Table 24. Cohen's kappa scores for character level perturbations in SS 46
Table 25. Cohen's kappa scores for character word perturbations in SS 47
Table 26. Cohen's kappa scores for ither perturbations in SS 48
Table 27. From left to right, average kappa score per model, per NLP task and per perturbation 49
Table 28. From left to right, average kappa score for character level, word level and other perturbations 50
Table 29. Average kappa score per type of perturbation when non-represented perturbations are removed 50

List of figures

Figure 1. Example of distributions shifts 8
Figure 2. Visual representation of attention and self-attention 13
Figure 3. The DistilBERT model and its components 17
Figure 4. The ELECTRA model and its components 18
Figure 5. The XLNet model and its components 18
Figure 6. The funnel transformer model and its components 19
Figure 7. Confusion matrix to explain the computation of the Kappa Cohen's score 29
Figure 8. Cohen's kappa evaluation for character level perturbations of ELECTRA in GC 36
Figure 9. Cohen's kappa evaluation for word level perturbations of DistilBERT in GC 37
Figure 10. Cohen's kappa evaluation for character level perturbations of XLNet in HSOL 40
Figure 11. Cohen's kappa evaluation for character level perturbations of XLNet in NLI 42
Figure 12. Cohen's kappa evaluation for character level perturbations of DistilBERT in SA 44
Figure 13. Cohen's kappa evaluation for character level perturbations of XLNet in SS 47

1. Introduction

Data Science and related technologies are constantly evolving, and especially in the area of Machine Learning (ML), and Artificial Intelligence (AI) more broadly. What began with feed-forward neural networks has rapidly developed in new ways of processing, computing, and understanding large amounts of data in order to extract patterns and aid in decision-making.

This paradigm has led to significant advances in multiple fields, including healthcare (Yu, Beam, & Kohane, 2018), marketing (Brei, 2020) and computer vision (Voulodimos, Doulamis, Doulamis, Protopapadakis, & others, 2018), among others. One of the key areas where the most remarkable improvements have been made is Natural Language Processing (NLP), which is the domain of processing, interpreting, and comprehending natural language (Reshamwala, Mishra, & Prajakta, 2013). The huge success in this discipline has contributed to lowering the barrier for some complex tasks, such as programming, with popular models like GPT-3 (Brown, et al., 2020) and GPT-4 (OpenAI, GPT-4 Technical Report, 2023) (and their implementations in ChatGPT (OpenAI, Introducing ChatGPT, 2022)) allowing for all kinds of users to perform certain specific tasks without having that much expertise using natural and comprehensive language.

Pre-trained Large Language Models (LLMs) in Natural Language Processing have transformed the field by providing models that can automatically learn from large amounts of raw data in a self-supervised manner, generalise well, and perform efficiently in various downstream NLP tasks. Pre-trained LLMs, such as BERT and GPT, are characterised by their ability to extract representational structures from raw text data that encapsulate semantic and syntactic information, producing coherent output, such as grammatically correct text. These models have undoubtedly advanced the state of the art in NLP tasks such as language translation, sentiment analysis, question answering and speech recognition, which are critical in applications such as customer service, marketing and healthcare.

However, even though pre-trained LLMs promise efficient and optimal performance in most NLP tasks, there are still challenges that need to be addressed. It is usually the case that LLMs rely on well-curated and pre-processed datasets that do not reflect the noisy and diverse nature of real-world text. These models may thus be susceptible to

perturbations that can affect their performance. Perturbations can appear in all shapes and forms: they can happen at a character level, where individual characters are inserted, deleted, or replaced by others; they can happen at a word level, where for instance a word is replaced by a synonym; and finally, there are other types of perturbations, like common expressions being contracted (they are → they're). Depending on the model or the tokenisation process, some types of perturbations may affect the models more than others. Perturbed inputs can thus range from something as simple as a typo to completely out-of-vocabulary (OOV) words or phrases. A good LLM should be able to process and understand these perturbations and produce meaningful outputs. The specific tasks LLM are trained on can also affect the impact of the perturbations: some tasks may rely more on specific details in the words, and as such character level perturbations may be more detrimental, while others may have more trouble generalising concepts, so word level perturbation such as synonym replacement may change the output more drastically.

A good LLM should be able to process and understand these perturbations and produce meaningful outputs. Therefore, understanding the robustness of LLMs to distributional shifts and perturbations in the data is of paramount importance. This is essential to ensure that these models maintain their efficiency and accuracy when used in real-world applications.

This work addresses these challenges, focusing on the impact of perturbations on LLMs. We will discuss how LLMs are pre-processed and curated to reduce noise and erroneous inputs that lead to poor performance in real-world environments. We will also examine how specific language tasks can affect LLM performance in the presence of perturbations. We will also present approaches and methodologies to evaluate the robustness of LLMs.

1.1. Motivation

The increasing use of large language models for various applications and domains makes it crucial to assess their robustness under different data distribution shifts. To achieve this, it is essential to investigate how different algorithms and models are affected by these variations.

Current evaluation methods assess the offline performance metrics of models based on held-out test sets, typically with the same distribution as the training data. However, in real-world scenarios, the distribution often shifts between the development and deployment environments, leading to significant performance impacts. Therefore, assessing the robustness of a model to distribution shifts becomes critical.

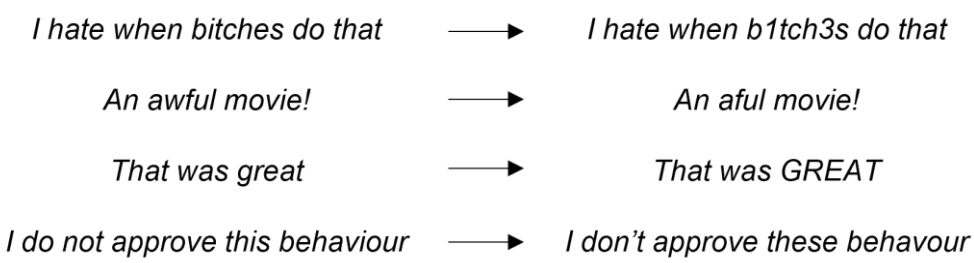

Figure 1. example of distributions shifts. In controlled environments, typos and perturbations are rare: however, in real world contexts, they are really common, producing shifts in data distributions

This work addresses this challenge by analysing the effects of ML algorithm, task, perturbation/transformation on model robustness. We focus on common NLP tasks like grammatical coherence, hate speech and offensive language, natural language inference, sentiment analysis and semantic similarity, and evaluate model robustness in the context of distribution shifts induced by transformation-based perturbations. Overall, this project aims to contribute to improve reliability and performance of large language models, which will benefit their widespread use in various application areas.

1.2. Objectives

The aim of this project is to perform a comprehensive evaluation of the robustness of different large language models (LLMs) under different types of perturbed inputs. For this purpose, we will fine-tune four LLMs on five different tasks, predicting both perturbed and unperturbed sentences. By measuring the difference between the predictions made on perturbed and unperturbed sentences, we will compare the robustness of different models.

Our project aims to address three key research questions related to the robustness of LLMs:

1. Which LLMs are more robust, and which are more vulnerable to perturbed inputs?
2. Which NLP tasks are more sensitive to perturbations? Understanding the effect of perturbations on different NLP tasks can help identify task-specific vulnerabilities and inform the choice of robust models for specific applications.
3. Which perturbation types and their level of accumulation (characters, words, or other) have a greater impact on model predictions? A better understanding of the perturbations that have a greater impact on model predictions can further help to refine evaluation metrics and develop more robust models.

To achieve our research objectives, we will conduct extensive experimentation and evaluation using various benchmark datasets and state-of-the-art LLM algorithms. Our research will use a diverse set of perturbation methods, including insertion, deletion, synonym substitution, among others, to simulate distribution shifts and evaluate model robustness under these scenarios.

1.3. Expected impact

The impact of this research project lies in the effect of text perturbations on large language models robustness. Understanding which perturbations affect models the most can lead to the development of new models or the improvement of existing ones with higher robustness and better performance even in the face of corrupted inputs.

In addition, this work could have real-world applications in tasks such as machine translation, sentiment analysis and text classification. Identifying which types of perturbations have the greatest impact on these tasks could lead to the development of targeted methods for data augmentation or model training, thereby improving accuracy and efficiency. By gaining a deeper understanding of the inner workings of NLP models, this work could also pave the way for future innovations and advances in the field. For example, the insights gained from this research could inspire the development of new architectures or pre-training approaches that deal with the perturbation types to which models are most susceptible.

1.4. Ethical and legal implications

Regarding data protection and availability, all the models and datasets were obtained from HugginFace (Wolf, et al., 2020), a company that aims to democratise AI through open source, so all datasets and models are open data. Furthermore, all data generated in this project will also be publicly available and free.

The relationship between this project and the Sustainable Development Goals can be found in Annex I.

2. State of the art and background

The history of Large Language Models (LLMs) in Natural Language Processing (NLP) has witnessed significant advancements over the years, driven by breakthroughs in machine learning and the availability of vast amounts of textual data.

In the beginning, NLP was approached in the same as fashion as other Machine Learning problems that required the analysis of sequences: Hidden Markov Models (Rabiner & Juang, 1986) (HMM) in combination with n-grams (sequence of characters or words of length n) were used to determine the probability of a sequence appearing given a certain distribution that could be derived from some data. These models had limited context awareness and struggled with capturing complex linguistic patterns.

Later, with the rise of neural networks, more complex models than HMM were developed: the neural language model was born (Bengio, Ducharme, & Vincent, 2000), which used a neural network architecture to capture dependencies between words and generate more fluent text. However, at the time these types of models required excessive data and computation to yield significant results.

Recurrent Neural Networks (RNNs) became popular for language modelling due to their ability to capture sequential dependencies in text. In 2010, Mikolov et al. introduced the Recurrent Neural Network Language Model (Mikolov, Karafiát, Burget, Cernocký, & Khudanpur, 2010), which achieved improved performance in generating coherent and contextually appropriate text. However, RNN-based models still faced challenges with long-term dependencies and suffered from vanishing or exploding gradients.

Some of these problems were mitigated by specific types of RNNs such as Long Short-Term Memories (Hochreiter & Schmidhuber, 1997) (LSTM) and Gated Recurrent Units (Cho, et al., 2014) (GRU). Finally, the encoder-decoder architecture (Sutskever, Vinyals, & Le, 2014), in conjunction with attention mechanisms (Vaswani, et al., 2017), gave birth to the transformer (Vaswani, et al., 2017), the most used type of model nowadays.

LLMs are usually pretrained in the following fashion: first, the sentence is divided in a process called tokenisation, which consists of breaking down the input into smaller parts, usually words, subwords or even characters, called “tokens”. Next, the sequence of

tokens is usually normalised and processed, with operations such as lowercasing if needed or removing punctuation and stop words. Finally, these curated tokens are converted into numerical vectors that try to capture the context of words (for example, words like “house” and “home” may have similar representations in these numerical spaces). A popular technique to discover these representations is embeddings (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013).

As such, NLP models take sequences as input: hence, the order of the series is important. The most common architecture used in LLMs is the encoder-decoder, which is divided into two parts. The encoder receives a variable-length input sequence and returns a fixed length vector that tries to capture the contextual information of the sentence. The decoder then takes this context vector and outputs a variable-length sequence that can be reconverted into tokens and thus an output sentence. This framework is used in sequence-to-sequence tasks, such as text summarisation, text generation or translation.

There is more to NLP models: for instance, the encoder-decoder architecture alone is not able to handle long sentences, where the dependencies and relationships between words in the text may be distant, and so, other mechanisms such as attention and self-attention (Vaswani, et al., 2017) complement encoder-decoder by mitigating this problem.

Broadly speaking, attention allows the model to determine, for a given token, which other parts in the sentence are more related to that token. For example, in Spanish, the word “las” may be more related to the word “comen”, since there is a concordance in number. Self-attention is used within the same sentence, while attention helps determine which parts of the encoded input are more relevant for predicting the token in the decoded output:

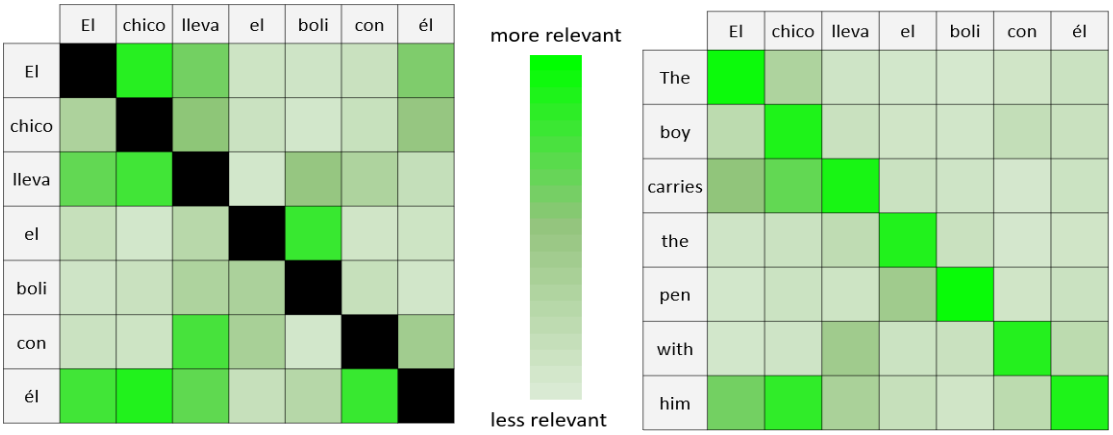

Figure 2. on the left, visual representation of self-attention. On the right, visual representation of attention. Attention matrices help visualise syntactic relationships, such as the dependency of the subject ("el chico") and their passive mention ("con él")

Figure 1 illustrates how attention and self-attention function. For each row, in the matrices, a greener colour indicates more relevance: for example, the words “chico” and “El” are the most important to predict the word “lleva”, since they dictate the number of the subject (singular). This in an example of self-attention. For attention, the words “El”, “chico” and “él” help the model the most to predict the word “him”, mainly because the word “him” refers to the subject of the sentence (“el chico” – “él”).

This mechanism, in conjunction with the encoder-decoder architecture and some additional features to help improve the efficiency of the model like positional encoding, form the most popular type of NLP model nowadays: the transformer.

Transformers are pretrained to generate outputs in the form of sequences: that is, they do not generate a single label or number, but rather, as discussed earlier, an ordered sequence of tokens. However, they can be finetuned in specific tasks to produce any kind of output, be it sequences or single outputs[1]. This flexibility makes NLP models useful and applicable in a variety of domains, especially the ones where unstructured data is common, such as in medicine and disease diagnosis.

One state of the art example of a transformer is Bidirectional Encoder Representations from Transformers, or for short, BERT (Devlin, Chang, Lee, & Toutanova, 2019). This model played a pivotal point in shaping the NLP landscape,

[1] This is usually accomplished by replacing the last layer of the decoder on the model by a fully connected layer whose inputs are the hidden states of the decoder.

proving the efficacy of the transformer architecture and popularising it. BERT uses a pretraining approach called masked language modelling (MLM), which consists of hiding (masking) some of the tokens to the transformer. Then the model must predict the sentence and those masked inputs.

This technique helps the model learn the intricacies of the language. However, it ignores the dependency between the masked inputs. For instance, a sentence like “I’m going to New York” with two masked tokens could be “I’m going to [MASK] [MASK]”. BERT does not retain any dependencies between the two words, so a valid (and more probable that “New York”) prediction would be “I’m going to Paris [MASK]” → “I’m going to Paris tomorrow”.

This is the general outlook of modern large language models, and NLP is constantly evolving with the proposal of new techniques and approaches. However, even though the capabilities of modern artificial intelligence are remarkable, they can underperform with noisy data in real world contexts.

The discrepancy in performance of NLP models between benchmarks and real word applications has been discussed in other works (Ribeiro, Wu, Guestrin, & Singh, 2020), with the results indicating that there is a significant gap, where sentences with typos that are insignificant for humans are difficult for LLMs.

There is a distinction to be made between natural noise and synthetic noise in NLP: natural noise is derived from perturbations that can occur organically in real world environments, such as common typos and contractions found in raw datasets. Synthetic noise, on the other hand, refers to artificially produced perturbations, usually with the purpose of evaluating the robustness of NLP models. This synthetic noise can be used to create adversarial examples with tools like OpenAttack (Zeng, et al., 2021). Both types of noise have been proven to greatly impact predictions in some tasks like translation (Belinkov & Bisk, 2018).

However, despite the availability of technology to generate synthetic noise, its utilisation in benchmarks remains limited, and applying these perturbations could complement and improve them (Moradi & Samwald, 2021).

Moradi and Samwald (2021) studied the performance of different LLMs for different types of perturbations, specifically at the character and word level. They reported a

decrease in performance between 8 and 18 points[2], depending on the model, the task, and the number of perturbations per sample, concluding that models are more sensitive to character level perturbations than word level ones.

However, there a few considerations to take into account: firstly, other types of perturbations were not employed, such as contractions, additions of punctuation or more specific ones like addition of links or usernames similar to those founds on social media (like "@username789"). Moreover, the hyperparameter Perturbations Per Sample (PPS) is a flat number of perturbations allowed in a sentence. This choice can scale poorly with inputs of significantly varying length.

Furthermore, it is worth noting that due to the potential alteration of input meaning caused by certain word perturbations, the authors opted to manually select up to 200 perturbed samples to evaluate on the test size. This could affect the previous conclusion regarding character level perturbations being more impactful than word level ones, since character level ones did not pose this additional challenge. Enhancing the size of the test datasets could significantly bolster the validity of the obtained results.

Therefore, despite the existence of previous work regarding the evaluation of model robustness against perturbations, the research is not complete, and some additions and improvements can be made to ascertain previous results and discover new ones.

[2] "Points" indicating not percentual, but absolute decrease in the task chosen metric (for example, a decrease of 10 points in a task that uses accuracy as the main evaluation metric would imply a 0.1 decrease in accuracy)

3. Methodology

In order to achieve the stated objectives, we chose and followed a comprehensive methodology, which involved finetuning four different large language models on five distinct natural language processing (NLP) tasks.

Firstly, the models were subjected to finetuning and hyperparameter optimisation to ensure optimal performance. Then, this dataset was perturbed by different types of perturbations and was used to get alternative predictions. Two sets of predictions were obtained: one from the original, unaltered test dataset and the other from the same dataset subjected to different types of perturbations. These perturbations ranged from simple changes in words to more complex grammatical and semantic alterations. Then we measured the degree of disagreement between the two sets of predictions as an indicative of the model's overall robustness. To achieve this, a detailed evaluation process was implemented.

In what follows, each phase of the workflow will be described in detail, mainly the datasets, models and perturbations we used, a description of the finetuning process and how the main objectives were evaluated.

3.1. Models

Some of the models chosen are inspired by the previously described BERT but provide different alternatives, either to the original architecture or the pretraining approach, that circumvent these limitations. All models were obtained and finetuned via the HuggingFace transformers (Wolf, et al., 2020) library and the Python library for deep learning PyTorch (Paszke, et al., 2017).

In order to offer more comprehensive assessments of large language models, this study deliberately selected a diverse set of models that exhibited variations in their architecture, preprocessing methodologies, and pretraining techniques.

- **DistilBERT** (Sanh, Debut, Chaumond, & Wolf, 2019): a distilled version of BERT, designed to retain its language understanding capabilities while being smaller and

therefore less computationally expensive. This is achieved by adopting a teacher-student relationship, where the larger, more expensive model (the teacher), must educate the smaller model (the student) and transfer its knowledge. This technique is called knowledge distillation. The version "distilbert-base-cased"[3] was used in this work.

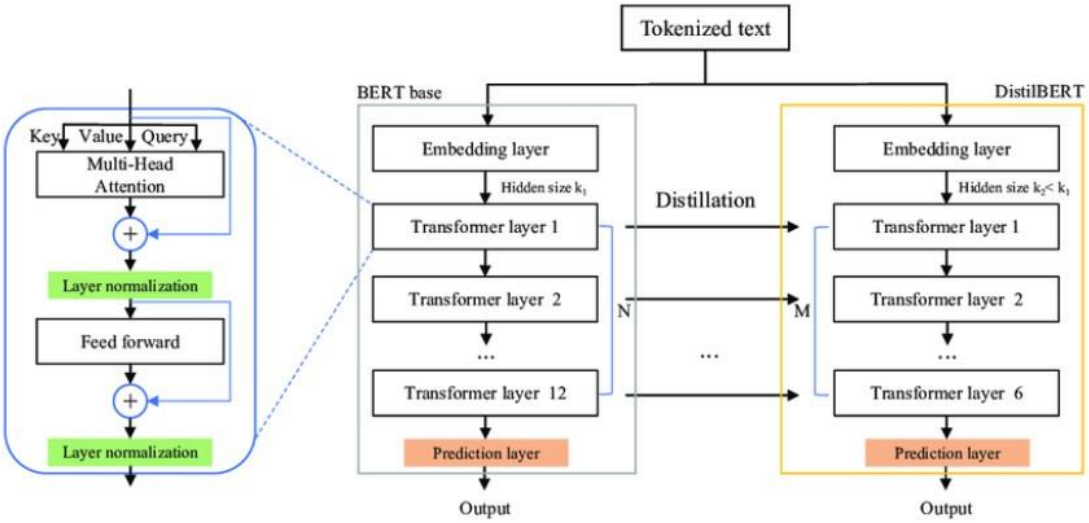

Figure 3. the DistilBERT model and its components. Source: (Adel, et al., 2022)

- **ELECTRA** (Clark, Luong, Le, & Manning, 2020): another variant of BERT with a unique and interesting pretraining approach: two transformer models are used (the generator and the discriminator), and instead of masking some tokens of the input sentence, they are replaced by plausible alternatives provided by the generator. The discriminator then learns to determine which tokens were replaced by the generator (to discriminate). This pretraining approach is less demanding on resources and leads to equal or even better results than models like the original BERT or large-scale models like XLNet[4]. The version "google/electra-base-discriminator" was used in this work.

[3] "cased" means it distinguishes between lower case and upper-case characters.
[4] When using the same amount of compute as XLNet.

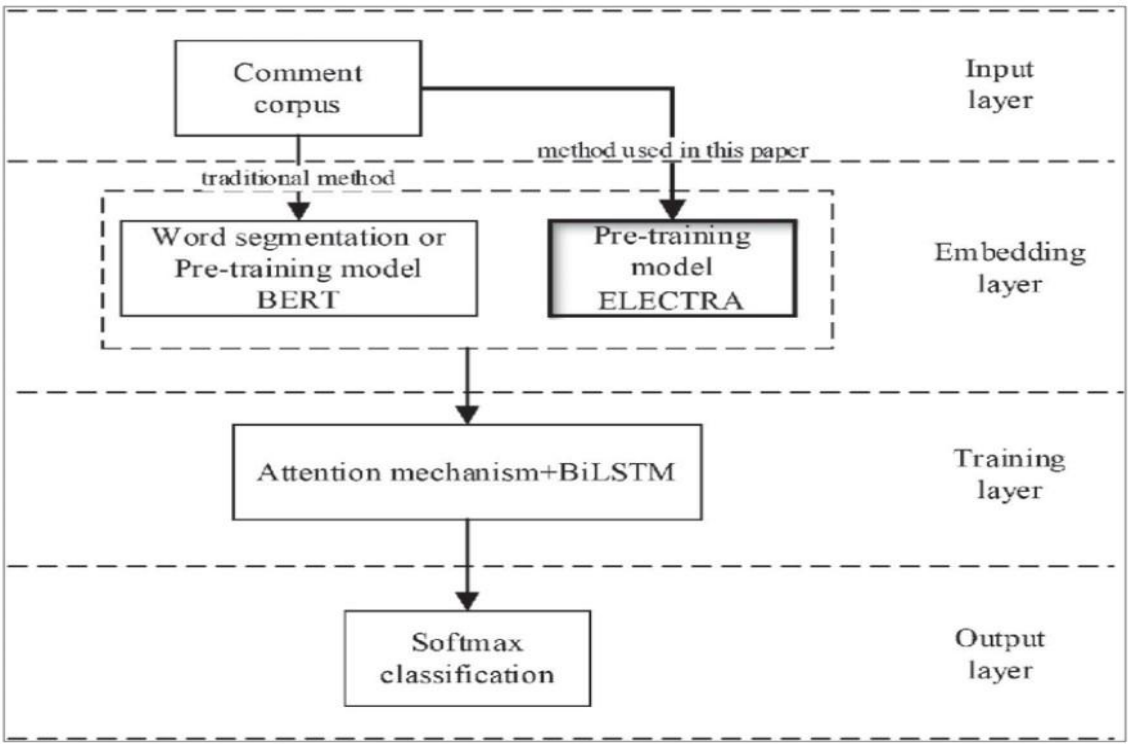

Figure 4. the ELECTRA model and its components. Source: (Zhang, Yu, & Zhu, 2021)

- **XLNet** (Yang, et al., 2019): an extension of the Transformer-XL (Dai, et al., 2019) model that, unlike BERT, does not neglect dependency between masked inputs, and by permuting the tokens of its inputs it learns bidirectionally. Given this pretraining approach, it is the larger model utilised in this work. The version "xlnet-base-cased" was used in this work.

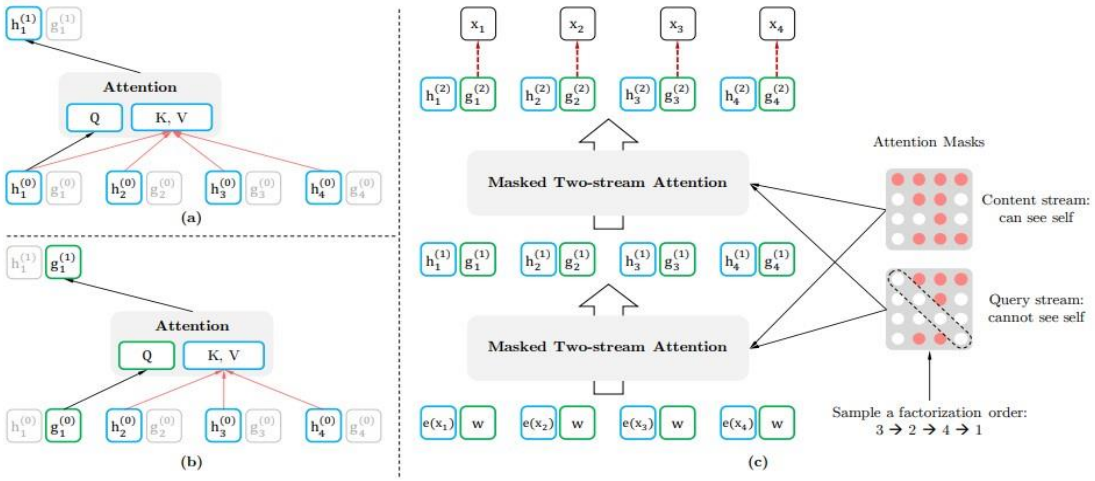

Figure 5. the XLNet model and its components. Source: (Yang, et al., 2019)

- **Funnel Transformer** (Zihang, Guokun, Yiming, & Quoc, 2020)**:** bidirectional transformer that uses a pooling[5] operation at the end of each block of layers, reducing its size and therefore the number of operations. If a sequence of the of a specific size is needed, there is an optional layer that upsamples the final hidden states. The version “funnel-transformer/small” was used in this work.

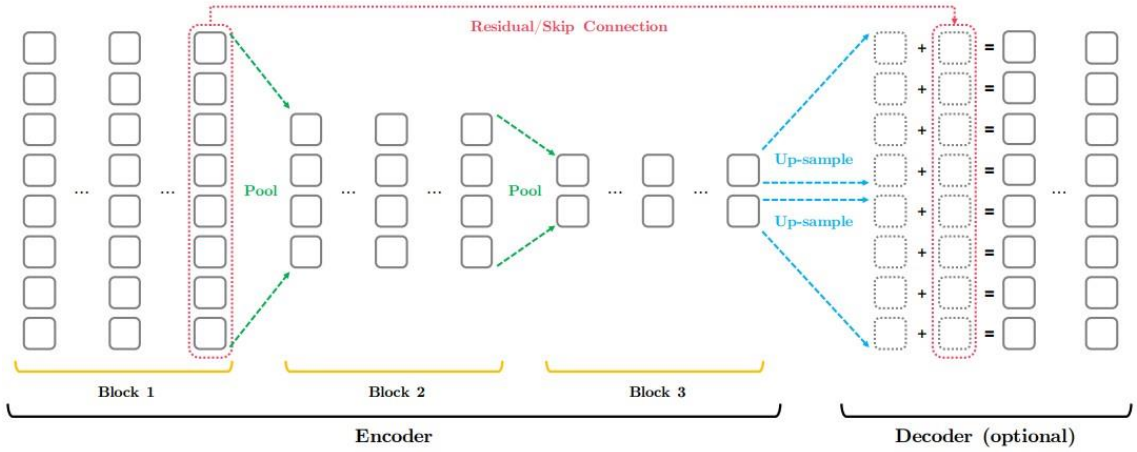

Figure 6. the funnel transformer model and its components. Source: (Zihang, Guokun, Yiming, & Quoc, 2020)

Basically, DistilBERT is a smaller and computationally cheaper version of BERT, achieved by using knowledge distillation through a teacher-student relationship. ELECTRA has a unique pre-training approach that uses two transformer models, a generator and a discriminator, to replace tokens with plausible alternatives, and trains the discriminator to determine which tokens have been replaced. XLNet pre-trains bidirectionally and does not neglect dependencies between masked inputs. Funnel Transformer uses a pooling operation to reduce its size and an optional upsampling layer to produce the desired sequence length. All models were extended with a final classification layer for single label classification tasks. A summary of the characteristics of each model is given in Table 1.

[5] Pooling is a technique mainly used in convolutional neural networks (CNN) that reduces the size of the output.

Model	Based of	Pretraining approach / Architecture	Computation cost	Year
DistilBERT	BERT	Knowledge distilation	Cheap	2019
ELECTRA	BERT	Discrimnation based of generator-discriminator models rather than generation	Cheap	2020
XLNet	Transformer-XL	Permutation of tokens	Expensive	2020
Funnel Transformer	-	Reducing size at each block by pooling	Cheap	2020

Table 1. Summary of the characteristics of the four models: DistilBERT, ELECTRA, XLnet and Funnel Transformer

For more information about specific parameters and hyperparameters such as the number of heads, dropout use, or size of input tokens, we refer to Annex II, although most values are the ones used by default.

3.2. NLP tasks

To ensure that the discoveries of this work are applicable to a multitude of fields in NLP, a selection of five different NLP tasks were chosen. These five tasks are supervised (classification): the model predicts a categorical label for each example. All datasets are in English and, like the models, were obtained via HuggingFace:

- **Sentiment Analysis (SA):** a task to determine the sentiment or opinion about a topic given a text. The output can be binary (like vs do not like) or ordinal, such as rating from 1 to 5. For this task, the Stanford Sentiment Treebank (Socher, et al., 2013) was chosen, which consists of sentences from movie reviews, and it is a binary classification task. An example of this dataset would be:

Sentence	Label
cold movie	0 (negative)
with his usual intelligence and subtlety	1 (positive)
will find little of interest in this film, which is often preachy and poorly acted	0 (negative)
a $ 40 million version of a game	0 (negative)
gorgeous and deceptively minimalist	1 (positive)

Table 2. Examples for the Stanford Sentiment Treebank with their corresponding label

- **Grammatical Acceptability (GA):** a task determining the grammatical validity of a sentence, which is really useful to assess if a model has grasped the rules of a language. For this work, the Corpus of Linguistic Acceptability (Warstadt, Singh, & Bowman, 2018) (CoLA) was chosen, which consists of sentences from books and journal articles on linguistic theory. It is also a binary classification task.

Sentence	Label
As you eat the most, you want the least	0 (not acceptable)
John was lots more obnoxious than Fred	1 (acceptable)
The tube was escaped by gas	0 (not acceptable)
We want John to win	1 (acceptable)
We persuaded Mary to leave and Sue to stay	1 (acceptable)

Table 3. Examples for the Corpus of Linguistic Acceptability with their corresponding label

- **Semantic similarity (SS):** task to determine the likeliness of similarity between sentences or texts. In this case, the Paraphrase Adversaries from Word Scrambling (Zhang, Baldridge, & He, 2019) (PAWS) dataset was utilised and consists of pairs of paraphrase sentences from Wikipedia generated by word swapping and back translation methods. Like the previous tasks, this is a binary classification task.

Sentence 1	Sentence 2	Label
It is the seat of Zerendi District in Akmola Region	It is the seat of the district of Zerendi in Akmola region	1 (Semantically similar)
BA transferred the former BCal routes to Heathrow to Tokyo and Saudi Arabia	BA relocated the former BCal routes to Tokyo and Saudi Arabia to Heathrow	0 (Not semantically similar)
He is trained by Andre Rozier and shares a gym with the former World Champion Daniel Jacobs	He is trained by Daniel Jacobs and shares a gym with former world champion Andre Rozier	0 (Not semantically similar)
The Tabaci River is a tributary of the River Leurda in Romania	The Leurda River is a tributary of the River Tabaci in Romania	0 (Not semantically similar)
Steam can also be used , and does not need to be pumped	Also steam can be used and need not be pumped	1 (Semantically similar)

Table 4. Examples of the Paraphrase Adversaries from Word Scrambling dataset with their corresponding label

- **Natural Language Inference (NLI):** task about logical inference relation of a pair of sentences: either entailment, contradiction or neither (neutral). We used the Multi-Genre Natural Language Inference Corpus (Williams, Nangia, & Bowman, 2018) (MNLI), which consists of pair of sentences from different sources such as transcribed speech, fiction, and government reports. Contrary to the previous tasks, there are three classes instead of two.

Premise	Hypothesis	Label
Gays and lesbians	Heterosexuals	2 (Contradiction)
yeah i mean just when uh the they military paid for her education	The military didn't pay for her education	2 (Contradiction)
(Read for Slate's take on Jackson's findings)	Slate had an opinion on Jackson's findings	0 (Entailment)
yeah well you're a student right	Well you're a mechanics student right?	1 (Neutral)
Well you're a mechanics student right?	yeah well you're a student right	0 (Entailment)

Table 5. Examples of the Multi-Genre Natural Language Inference Corpus dataset with their corresponding label

- **Hate Speech and Offensive Language (HSOL):** task to determine whether a sentence or comment (most datasets are created from social media posts) can be considered as hate speech or offensive language. In this case, the dataset employed in the paper “Automated Hate Speech Detection and the Problem of Offensive Language” (Davidson, Warmsley, Macy, & Weber, 2017) was used, which consists of tweets with three possible labels: hate speech, offensive language or neither.

Sentece	Label
@rhythmixx_ :hobbies include: fighting Mariam bitch	1 (Offensive langauge)
@AllAboutManFeet: http://t.co/3gzUpfuMev woof woof and hot soles	2 (Neither)
@CB_Baby24: @white_thunduh alsarabsss hes a beaner smh you can tell hes a mexican	0 (Hate Speech)
@CauseWereGuys: Going back to school sucks more dick than the hoes who attend it	1 (Offensive langauge)
@ArizonasFinest6: Why the eggplant emoji doe? y he say she looked like scream lmao	2 (Neither)

Table 6. Examples of the Multi-Genre Natural Language Inference Corpus dataset with their corresponding label

A brief summary of the five datasets used can be found in Table 7. Overall, the selection of these five NLP tasks ensures that the findings of this study can be applied to various domains.

Dataset	NLP Task	Size	Nº labels
CoLA	Grammatical Acceptability	Train: 8 550 sentences Validation: 1 040 sentences Test: 1 060 sentences	2
HSOL	Hate Speech and Offensive Language	Train: 13 878 tweets Validation: 5 948 tweets Test: 4 957 tweets	3
MNLI	Natural Language Inference	Train: 15 000 pairs of senteces Validation: 9 820 pairs of sentences Test: 9800 pairs of sentences	3
SST2	Sentiment Analysis	Train: 15 000 sentences Validation: 872 sentences Test: 1820 sentences	2
PAWS	Semantic Similarity	Train: 15 000 pairs of senteces Validation: 8 000 pairs of sentences Test: 8 000 paris of sentences	2

Table 7. Summary of the five datasets: CoLA, HSOL; MNLI, SST2 and PAWS

To improve the speed of the finetuning phase, datasets with more than 15,000 examples in the training set were reduced to that number of sentences. Validation and test sets were not altered since their size never exceeded this number.

In the case of the HSOL dataset, only the train set was provided, so the data was partitioned with stratification into train-validation-test (55-25-20). To find more information and details about the datasets, refer to Annex II.

3.3. Perturbations

There were three different types of perturbation levels used: character-level perturbations (inserting, deleting or substituting a character in the sentence), word-level ones (substituting a word in the sentence), and other types of perturbation (adding punctuation marks, applying common contractions, inserting adverbs, etc.). These types will be evaluated separately.

3.3.1. Character-level perturbations:

- **Keyboard**: replaces characters with other characters that are close in a keyboard with QWERTY layout.
- **OCR:** replaces characters to simulate Optical Character Recognition (OCR) mistakes.
- **Spelling error:** transformation that leverages a predefined spelling mistake dictionary to simulate common spelling mistakes.
- **Typos**: randomly inserts, deletes, swaps or replaces characters.

Instead of the Perturbations Per Sample (PPS) hyperparameter (Moradi & Samwald, 2021) we use the percentage of perturbation of the sample: for instance, a sentence with 20 characters and a character perturbation percentage of 10% should have around 2 characters affected. Using percentages instead of absolute values helps scale the perturbation effect to variable-length sentences. The values used for character perturbation percentage were 1%, 5% and 10%.

3.3.2. Word-level perturbations

- **SwapSynWordEmbedding:** replaces a word with a synonym derived from the GloVE (Pennington, Socher, & Manning, 2014) (word embeddings).

- **SwapSynWordNet:** replaces a word with a synonym derived from the WordNet (Fellbaum, 1998) project.

Similar to character-level perturbations, there is a hyperparameter named word perturbation percentage: a sentence with 10 words and a word perturbation percentage of 10% should have the 1 word altered. Since sentences in the datasets used have from 5 to 20 words, the values for this hyperparameter were 10%, 20% and 30%.

3.3.3. Other perturbations

- **Contraction:** replaces phrases like “will not” and “he has” with contracted forms, namely, “won’t” and “he’s”.

- **InsertAdv:** adds and adverb before a verb.

- **Prejudice:** replaces a location (a country, a city, etc.) with another location that could cause prejudice or bias (such as a city or a country in a poor area).

- **Punctuation:** adds punctuation at the end of the sentence.

- **SwapNum:** replaces digit numbers (4, 10, 1048) by other numbers.

- **VerbTense:** changes the tense of a verb in a sentence.

- **Twitter:** adds mentions to usernames (@username789) or short links (http://t.co/blS7dalMiF) similar to those found in Twitter.

- **WordCase:** changes all the characters in a sentence to be upper case.

It is clear from the above that even for a perfectly robust models some perturbations should change the output (e.g., a question that depends on the numbers when using SwapNum, such as “this film scores 10 out 10” instead of “this film scores 0 out of 10” may affect sentiment although not grammatical acceptability). The incorporation of perturbations into the models was carried out using the powerful Python library textflint (Wang, et al., 2021). This library was developed specifically for evaluating the robustness of natural language processing and provides a variety of techniques for each of the perturbations mentioned.

A summary and an example of each perturbation can be found in Table 8.

Level	Perturbation	Original	Perturbed
Character	Keyboard	hand	hanf
	Ocr	hello	hellu
	SpellingError	achieve	acheive
	Typos	random	andom
Word	SwapSynWordEmbedding	It is nothing but a caricature	It is nothing but an imitation
	SwapSynWordNet	I'd like some pie, please	I'd like some cake, please
Other	Contraction	They are coming, run!	They're coming, run!
	InsertAdv	He jumped the fence	He quickly jumped the fence
	Prejudice	I'm moving to Canada next month	I'm moving to Morocco next month
	Punctuation	The film was quite good	The film was quite good!
	SwapNum	3 people attended the meeting	15 people attended the metting
	VerbTense	He started slow but ended up winning	He starts slow but ends up winning
	Twitter	Good premise, bad ending	Good premise, bad ending @username789
	WordCase	I'm not angry, not at all	I'M NOT ANGRY, NOT AT ALL

Table 8. Summary of the perturbations with a short example

It is likely that some of these transformations are not applicable to all sentences: for instance, not all texts have digit numbers in them, so the perturbation **SwapNum** will not perturb those sentences. For that reason, after each sample is perturbed, it is checked whether this new perturbed sample is actually different from the original one, and a tally is made. When the whole dataset is processed, the percentage of total perturbed samples is returned. This value can be used to ascertain the level of validity of the results, since very few conclusions can be made if the dataset was perturbed, for example, only by 10%. This also gives some insight into how frequent that perturbation can appear in real contexts.

It is important to note that, since some types of perturbations can lead to logical changes in the outcome (e.g., replacing a word with its antonym can change a positive review into a negative review), the main objective is not to measure the performance degradation (such as drop in accuracy) of these models, but rather the number of changed predictions.

3.4. The finetuning process: hyperparameter optimisation

Even though the models we have used have great performance on their own, their real potential can be attained only if they are finetuned for a specific dataset or task. Therefore, every model was trained on each one of the previously mentioned datasets, giving a total of 20 training processes.

For each of these processes, some level of hyperparameter optimisation was performed: specifically, the hyperparameters that were altered were the initial learning rate (all models use a learning rate scheduler with linear decay each epoch) and the number of epochs. The values for the initial learning rate were 0.00001, 0.0005 and 0.0001. The models were trained for 5, 7 and 10 epochs. This means every model was trained 9 different times, leading up to 180 different training sessions.

Even though more hyperparameters could have been tuned, the number of training sessions increases drastically for every new combination of them. Considering more hyperparameters turned out to be prohibitive for the time frame of the project.

The models were trained using the training partition of the datasets. At the end of each training session, the accuracy and f1 score of both the training and validation sets were computed, and the model checkpoint (weights, biases, random states, etc.) was saved for later use in the evaluation phase.

The score for determining which combination of hyperparameters is better was obtained using the following formula:

$$score = 0.2 \cdot f1_{train} + 0.8 \cdot f1_{validation}$$

This custom calculation was chosen because in rare occasions, especially when training on a low number of epochs, the f1 score of the validation dataset can be higher due to pure randomness, distorting the real capability of the model. Furthermore, even though the accuracy and f1-score are calculated, this custom score only employs the f1 score due to being a more robust metric, effective against imbalanced datasets, contrary to the accuracy, which is highly sensitive to asymmetric class distributions.

All training processes used a batch size of 32, except XLNet sessions, which, due to the limited size of the GPU, used a batch size of 16.

Hyperparameter	Value
Batch size	32 (16 XLNet)
Initial learning rate	0.0001 / 0.0005 / 0.00001
Learning rate decay	Linear decay per epoch
# of epochs	5 / 7 / 10
Optimiser	Adam with weight decay (AdamW)
Weight decay	0.01
Adam β_1	0.9
Adam β_2	0.999
Adam ε	1e-8

Table 9. Hyperparameters employed in the finetuning phase. The only two who take various values are the initial learning rate and the number of epochs

3.5. The evaluation process: perturbing and predicting sentences

In order to identify discrepancies in model predictions, the test portion of each dataset was subjected to a series of perturbations as outlined previously, resulting in a collection of perturbed datasets that can now be accessed on HuggingFace[6].

Once the best combination of hyperparameters was found, the models were loaded thanks to the checkpoint system previously mentioned. Each original (unaltered) test set was then fed into the model, returning some label predictions. As many of the datasets used in this study are also used in popular benchmarks such as the GLUE benchmark (Wang, et al., 2019), test labels were often unavailable, making it impossible to calculate

[6] https://huggingface.co/DaniFrame

evaluation metrics such as accuracy or f1 score. With this in mind, the perturbed datasets were loaded and fed into the model, generating alternative predictions for each perturbed dataset. To quantify the degree of disagreement between the two sets of predictions, Cohen's kappa coefficient (Cohen, 1960) was used, which accounts for the possibility of random disagreement. The Cohen's kappa formula is as follows:

$$\kappa = \frac{p_0 - p_e}{1 - p_e}$$

Where p_0 is the relative observed agreement among raters (accuracy), and p_e is the hypothetical probability of chance agreement. To give an example, given the following confusion matrix:

		Rater 1		
		Class A	Class B	Total
Rater 2	Class A	40	10	50
	Class B	20	30	50
	Total	60	40	100

Figure 7. Confusion matrix to explain the computation of the Kappa Cohen's score

The observed agreement among raters is:

$$p_0 = \frac{40 + 30}{100} = 0.7$$

To compute the probability of change agreement, note that Rater 1 predicted Class A 60 (40+20) out of 100 times (0.6), and Rater 2 predicted Class A 50 (40+10) out of 100 times (0.5). So, the chance that both raters predict Class A is 0.6 · 0.5 = 0.3.

Similarly, for Class B, the chance that both raters predict said class is:

$$p_B = \frac{40}{100} \cdot \frac{50}{100} = 0.4 \cdot 0.5 = 0.2$$

Finally, the change that the raters agree by chance in any class is 0.3 + 0.2 = 0.5. Applying the definition of the Cohen's kappa formula, we get that the kappa score is:

$$\kappa = \frac{0.7 - 0.5}{1 - 0.5} = 0.4$$

Values of Cohen's kappa coefficient close to 1 indicate a high degree of agreement between the predictions for most samples, while values close to -1 indicate the opposite. A value around 0 indicates that the number of prediction disagreements is as expected by chance. It should be noted that predictions are deterministic: as such, the Cohen's kappa between the predictions unperturbed and themselves is 1.

When aggregating results across datasets and perturbations, the proportion of perturbed examples in each perturbed dataset was taken into account to calculate the 'robustness score', rather than relying solely on Cohen's kappa coefficient. This was done to prevent models from receiving inflated scores when the dataset had minimal perturbations, such as in SwapNum.

3.6. Hardware, reproducibility and reporting results

Both finetuning and evaluation were performed in a server with an Intel® Core™ i9-10920X at 3.5 GHz, a 126 GB memory, and 2 NVIDIA GeForce RTX 3090 GPUs with 24GB of dedicated memory each. The perturbations were made in CPU while the finetuning and prediction were performed in GPU.

To ensure reproducibility and in compliance with the recommendations of the Science paper about reporting of evaluation results in AI (Burnell, et al., 2023), all finetuning and evaluation results are included in the following repository[7] to avoid recomputation and therefore avoidable energy consumption. Various seeds that control different random processes (mainly from the libraries transformers, PyTorch and NumPy) in the finetuning phase were fixed, but there are still some random processes in GPU that are not fixable, so upon reproducing the code results may vary slightly, although the conclusions should be the same. This repository also contains all the instances-level results of the finetuning phase, as well as the results for Cohen's kappa from the perturbations.

[7] https://github.com/Daniframe/TFG-GCD

4. Results

The following sections contain the different results for the finetuning phase as well as the evaluation phase. Let us remember that the three objectives this work depend on the variations of models, NLP tasks (represented by datasets) and perturbations.

4.1. Finetuning results: finding the best hyperparameters

		Initial learning rate		
		0.00001	0.00005	0.0001
Number of epochs	5	0.874630	0.881330	0.871070
	7	0.878898	0.879571	0.881094
	10	0.879833	**0.884099**	0.874800

Table 10. Example of the scores obtained for the finetuning phase in the case of finetuning DistilBERT for the Grammatical Coherence task. The highest score (optimal combination of hyperparameters) is highlighted

Table 10 illustrates the results of the fine-tuning phase. Each model-task pair (such as DistilBERT with Grammatical Coherence and ELECTRA with Sentiment Analysis) was evaluated with different hyperparameter combinations to determine their respective scores. The hyperparameter combination that produced the highest score was selected as the final model for the evaluation phase. Table 11 shows the results for all models and tasks, including the best number of epochs (left) and the optimal initial learning rate (right) for each pairing.

		NLP Task									
		GC (CoLA)		HSOL (HSO)		NLI (MNLI Corpus)		SA (SST2)		SS (PAWS)	
Model	DistilBERT	10	0.00005	7	0.00005	7	0.00005	7	0.00005	10	0.00005
	ELECTRA	10	0.00005	10	0.00005	5	0.00005	5	0.00005	7	0.00005
	Funnel Transformer	10	0.00001	5	0.00001	7	0.00001	10	0.00001	10	0.00005
	XLNet	10	0.00001	7	0.00005	10	0.00001	5	0.00001	5	0.00005

Table 11. Best number of epochs (left) and initial learning rate (right) for each combination of model and NLP task.

Several notable trends emerge from the results. The initial learning rate of 0.00005 generally produced the best results, while 0.0001 was never the optimal choice and sometimes produced vanishing or exploding gradients. It is also worth noting that the Grammatical Coherence task had the same number of training epochs for all models, highlighting its greater complexity compared to the other tasks.

In summary, the results vary depending on the hyperparameters, task and model. However, some trade-offs can be found. For instance, an initial learning rate between 0.00005 and 0.00001 seems to be the best value, since learning rates higher than that, such as 0.0001, lead to worse results overall.

4.2. Evaluation results

Although the fine-tuning process may not require extensive discussion, the perturbation evaluation process offers a wealth of analysis opportunities as it incorporates all three dimensions of the objectives of this project. In order to gain a full understanding of the impact of these perturbations and how they can affect the output of the model, we will examine several examples of these modified sets. We will then examine the results obtained from various model-task pairings, highlighting noteworthy details and attempting to explain them in terms of model or task characteristics. Finally, we will perform an aggregate analysis to determine which model, perturbation, and NLP task have the most significant reduction in robustness.

4.2.1. Some clarifying examples

To comprehend the final form of this analysis, a few examples will be shown next: the original and perturbed sample are contrasted, marking differences between them, and next to it the model prediction will be displayed. Remember that the main goal is not to measure whether this prediction is correct (performance), but rather if the model output changes at all (robustness, non-volatility). If the output changes, it indicates that the model is sensitive to small input variations.

These are some examples where altered sentences change the model's prediction:

Grammatical Coherence

John left orders for Bill not to leave (predicted as **grammatically coherent**)

John left ordsrs for Bill not to leave (predicted as **not grammatically coherent**)

Grammatical Coherence determines the grammatical validity of a sentence: that is, verbs tenses, gender to verb concordance, etc. But it should be able to ignore typos and obvious spelling mistakes. In this example, a human can derive from the rest of the sentence that "ordsrs" refers to "orders", and therefore the sentence is still grammatically correct. The model (in this instance DistilBERT) is not able to do this.

Hate Speech and Offensive Language

@Oskzilla go to sleep you fag! (predicted as **offensive language**)

@Oskzilla go to sleeping you fag! (predicted as **hate speech**)

In this case, a simple change in verb tense (which does not change the meaning of the sentence at all) is enough to change the label from offensive language (rude statements, profanities that can offend others) to hate speech (statements that promote discrimination against groups based on their race, gender, sexual orientation, etc.).

Natural Language Inference

Premise: Greuze gave a slow sigh

Hypothesis: Greuze jumped forth as a new idea sparked in his head

(predicted as **neutral**)

Premise: Greuze establish a slow sigh

Hypothesis: Greuze jumped forth as a new thought sparked in his head

(predicted as **entailment**)

Even though the coherence of the perturbed premise could be debated, it is undeniable that there is still no logical relationship between the premise and the hypothesis, and therefore a synonym via WordNet should not change the prediction from neutral to entailment.

Sentiment Analysis

vile and tacky are the two best adjectives to describe ghost ship (predicted as **positive**)

@nWV8 vile and tacky are the two best adjectives to describe ghost ship (predicted as **negative**)

The Twitter perturbation type is one of the most powerful to illustrate how brittle LLMs can be under specific conditions. The addition of noise like links or mentions to usernames adds nothing regarding a movie review, yet it can change the model's prediction. This is also an example where the perturbed phrase actually "helps" the model predict the correct label (since this review is evidently negative). Nonetheless, it still shows that the model (XLNet in this case) is volatile.

Semantic Similarity

Sentence 1: The Sunset Sunset Road comes from right and becomes Briscoe Mountain Road

Sentence 2: Sunset Road comes in from the right and becomes Briscoe Mountain Road

(predicted as **semantically similar** sentences)

Sentence 1: The Sunset Road cames from raight and becoms Briscoe Mountain Road

Sentence 2: Sunset Road comes in from the right and becomes Briscoe Montain Road

(predicted as **not semantically similar** sentences)

Another example where character perturbations alter the prediction. In this case, the words "comes", "right", "becomes" and "Mountain" contain easily spotted spelling mistakes that a human can recognise and correct, whereas for the model they change their output from semantically similar to not semantically similar.

These are some of the examples that show the results that will be analysed. Looking at specific instances can help understand whether the perturbations are applied in critical parts of the sentence, such as the subject or the verb, and to which degree humans can derive the original, unaltered sentence from the perturbed one.

4.2.2. Detailed analysis

Next, each NLP task will be discussed in detail, evaluating which model yielded better scores and which perturbations were more detrimental.

Grammatical Coherence

The results for the CoLA dataset can be found in Table 12.

Grammatical Coherence (CoLA) - Character level						
Perturbation	% of characters per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Keyboard	1%	0.41	**0.47**	0.37	0.43	100.0%
	5%	0.28	**0.30**	0.20	0.23	100.0%
	10%	0.08	**0.12**	0.05	0.06	100.0%
Ocr	1%	0.39	**0.44**	0.33	0.41	99.8%
	5%	0.21	**0.36**	0.25	0.29	99.8%
	10%	0.13	**0.17**	0.06	0.08	99.8%
SpellingError	1%	0.38	0.38	0.37	**0.39**	99.2%
	5%	**0.28**	0.26	0.18	0.27	99.2%
	10%	0.10	**0.10**	0.07	0.09	99.2%
Typos	1%	0.44	**0.48**	0.41	0.42	95.6%
	5%	0.32	**0.36**	0.24	0.32	97.6%
	10%	0.10	**0.16**	0.07	0.10	99.2%

Table 12. Cohen's kappas scores for each model and character level perturbation type for the Grammatical Coherence task. The best score for each perturbation is highlighted

As Table 12 shows, the scores are around 0.10 and 0.50, indicating an agreement between the predictions not produced by chance. However, this level of agreement is not very substantial, showing that character level perturbations do affect the robustness of the models, at least in this NLP task.

Something to highlight is that ELECTRA is generally more robust than the other models, especially in the Keyboard and Typos perturbations. ELECTRA also better

handles highly perturbed inputs, as the model has better scores than the other models when the percentage of altered characters is high (10%), contrary to Funnel Transformer, which is highly volatile in these conditions.

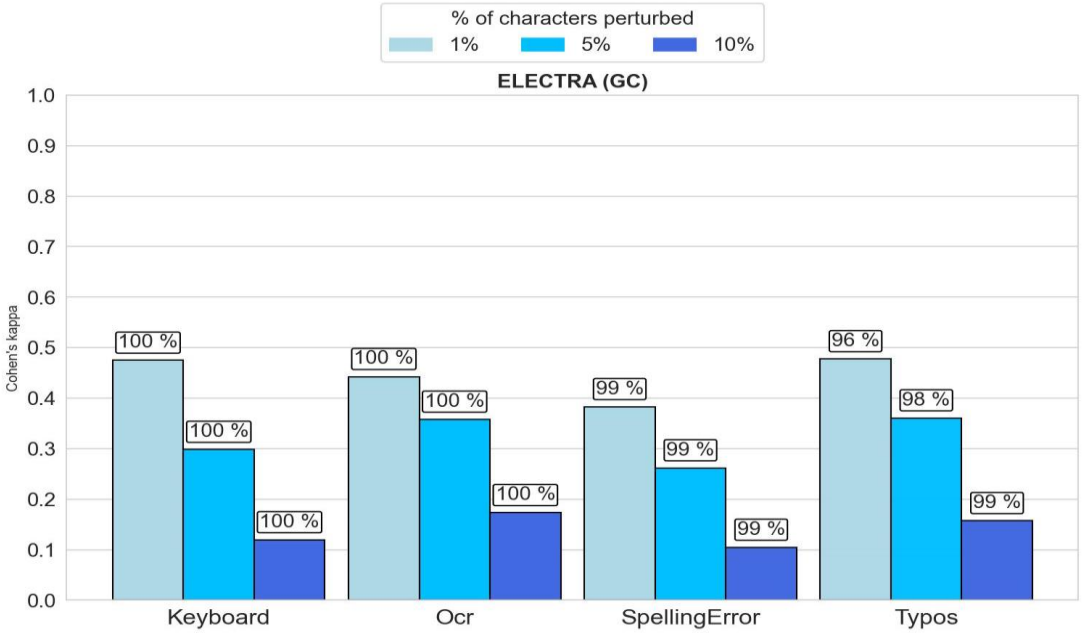

Figure 8. Cohen's kappa coefficient for character level perturbations in the finetuned ELECTRA model on the CoLA dataset. The value above the bars indicates the percentage of perturbed samples in the dataset.

Looking closely at the ELECTRA model, the relationship between the character perturbation percentage and the Cohen's kappa seems to be mostly linear, except in OCR, where the decrease from 1% to 5% is far less than the decrease from 5% to 10%, although more measurements should be made to support this claim.

Grammatical Coherence (CoLA) - Word level						
Perturbation	**% of words per sample perturbed**	**DistilBERT**	**ELECTRA**	**Funnel Transformer**	**XLNet**	**% of samples perturbed**
Swap Synonym Word Embedding	10%	0.89	**0.92**	0.87	0.87	39.7%
	20%	0.88	0.88	0.86	**0.89**	39.7%
	30%	0.88	0.89	0.88	**0.89**	39.7%
Swap Synonym WordNet	10%	**0.46**	0.32	0.32	0.37	99.8%
	20%	**0.39**	0.27	0.26	0.31	99.8%
	30%	**0.27**	0.16	0.16	0.15	99.8%

Table 13. Cohen's kappa scores for each model and word level perturbation type for the Grammatical Coherence task. The best score for each perturbation is highlighted

Regarding word level perturbations, Table 13 shows that ELECTRA does not perform as well as in character level perturbations. In this case, DistilBERT is the most robust model, with a kappa score that goes from 0.46 in low perturbed inputs to 0.27 in high perturbed ones.

The high scores near 0.9 in the Swap Synonym Word Embedding perturbation are due to the low proportion of examples perturbed in the dataset. This is why this metric was tracked, because it can uncover false good results: in this case, the high kappa values are because most examples were not perturbed at all, not altering the predictions.

Even though the scores when the perturbation percentage is low are similar to character level perturbations, the scores against highly perturbed inputs are much higher. This shows that models seem to be more sensitive towards character level perturbations.

As shown in Figure 3, the Cohen's kappa value does not seem to be influenced by the percentage of perturbed words in the case of Swap Synonym Embeddings, although this could be explained by the low number of altered samples. In the other perturbation, the relationship is not clear.

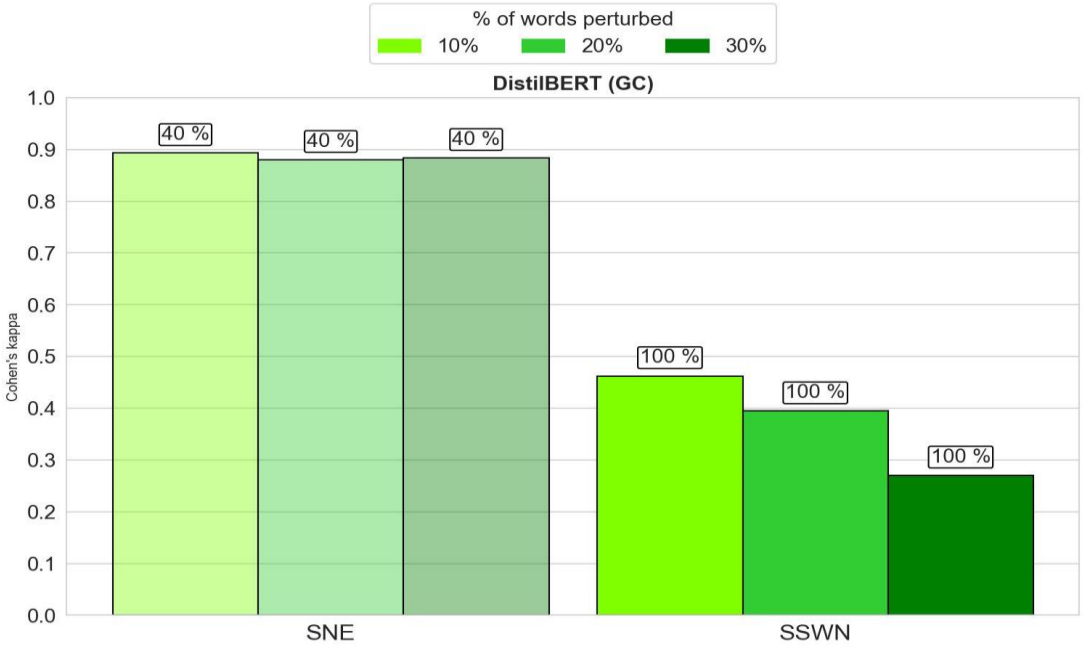

Figure 9. Cohen's kappa coefficient for word level perturbations in the finetuned DistilBERT model on the CoLA dataset. The value above the bars indicates the percentage of perturbed samples in the dataset

When analysing other types of perturbations, the results are much more mixed, especially due to some perturbations barely perturbing the sentences. Table 14 illustrates it perfectly, with perturbations such as Contraction, Prejudice and SwapNum having low proportions of perturbed samples. Discarding those, DistilBERT is also the most robust model, although ELECTRA and XLNet are close to it in some alterations like Contraction and Twitter.

Grammatical Coherence (CoLA) -Other					
Perturbation	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Contraction	**0.99**	0.98	0.96	0.98	16.7%
InsertAdv	**0.36**	0.24	0.22	0.30	99.4%
Prejudice	**1.00**	**1.00**	**1.00**	0.99	0.5%
Punctuation	0.84	**0.85**	0.76	0.57	100.0%
SwapNum	**1.00**	**1.00**	**1.00**	1.00	1.2%
VerbTense	**0.38**	0.23	0.31	0.35	88.1%
Twitter	**0.76**	0.74	0.64	0.76	100.0%
WordCase	0.00	**1.00**	**1.00**	0.18	100.0%

Table 14. Cohen's kappa scores for each model and other types of perturbations for the Grammatical Coherence task. The best score for each perturbation is highlighted

Another thing to consider is the perturbation WordCase, which changes all the characters in a sentence to be upper case. Notice how DistilBERT and XLNet have very low scores while ELECTRA and Funnel Transformer have perfect kappas (total agreement). This is probably due to the tokenisation process of the models: DistilBERT and XLNet distinguish between upper case and lower case characters, while ELECTRA and Funnel Transformer convert all characters to lower case. This might imply that case sensitive tokenisers are less robust, while models that do not differentiate between upper case and lower case can mitigate more effectively the impact of this type of perturbation.

The most detrimental alterations are InsertAdv and VerbTense, with scores between 0.22 and 0.38, while Punctuation and Twitter affect less all models.

Hate Speech and Offensive Language

Hate Speech and Offensive Language (hate_speech_offensive) - Character level						
Perturbation	% of characters per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Keyboard	1%	0.81	0.82	**0.84**	0.83	100.0%
	5%	0.62	0.62	0.65	**0.66**	100.0%
	10%	0.35	0.33	0.35	**0.40**	100.0%
Ocr	1%	0.83	0.81	0.82	**0.86**	100.0%
	5%	0.66	0.59	0.60	**0.72**	100.0%
	10%	0.41	0.31	0.31	**0.50**	100.0%
SpellingError	1%	0.79	0.80	**0.82**	0.81	98.5%
	5%	0.59	0.61	**0.64**	**0.64**	98.6%
	10%	0.38	0.41	0.45	**0.46**	98.6%
Typos	1%	0.83	0.81	0.83	**0.86**	97.0%
	5%	0.60	0.60	0.61	**0.64**	99.4%
	10%	0.36	0.29	0.34	**0.39**	99.9%

Table 15. Cohen's kappa for each model and character level perturbation type for the Hate Speech and Offensive Language task. The best score for each perturbation is highlighted

There is a clear contrast between Grammatical Coherence and Hate Speech and Offensive Language when it comes to Cohen's kappa values. While in the previous tasks the scores oscillated between 0.10 and 0.50, in HSOL the values are generally much higher, going from 0.3 at worst to almost 0.9 at best. Therefore, models are more robust against this task than Grammatical Coherence, probably because the language knowledge is not that demanding in HSOL.

The Cohen's kappa coefficients between models are more equal, but XLNet is the most robust model by a slight margin in almost every perturbation.

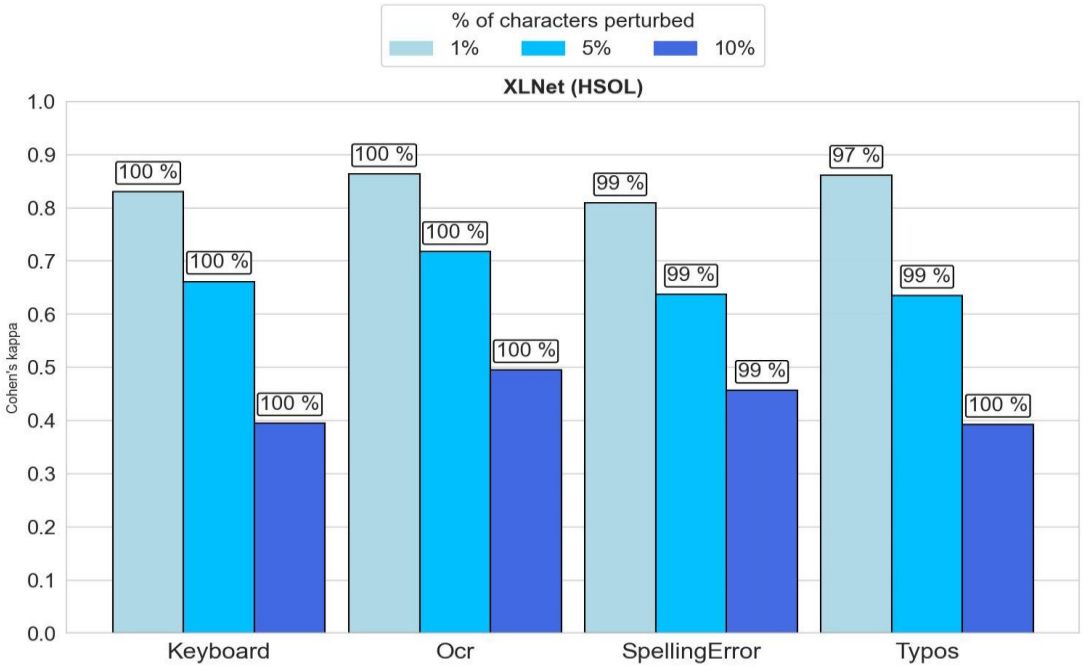

Figure 10. Cohen's kappa coefficient for character level perturbations in the finetuned XLNet model on the hate_speech_offensive dataset. The value above the bars indicates the percentage of perturbed samples in the dataset

Regarding word level perturbations, the same tendency as character level perturbations can be seen: all models present higher scores than in Grammatical Coherence. In this case, Funnel Transformer presents far better results than the rest of the models.

Hate Speech and Offensive Language (hate_speech_offensive) - Word level						
Perturbation	% of words per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Swap Synonym Word Embedding	10%	0.97	0.97	**0.98**	0.97	31.1%
	20%	0.97	0.97	**0.98**	0.98	31.1%
	30%	0.98	0.97	**0.98**	0.97	31.1%
Swap Synonym WordNet	10%	0.69	0.75	**0.78**	0.71	99.1%
	20%	0.55	0.62	**0.67**	0.58	99.1%
	30%	0.42	0.51	**0.56**	0.46	99.1%

Table 16. Cohen's kappa scores for each model and word level perturbation type for the Grammatical Coherence task. The best score for each perturbation is highlighted

This is also true for other types of perturbations, as shown in Table 17. In this case, there is not any transformation that greatly alters the predictions of the models, with values as high as 0.96.

Hate Speech and Offensive Language (hate_speech_offensive) - Other					
Perturbation	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Contraction	0.99	0.99	**1.00**	1.00	26.5%
InsertAdv	0.94	0.92	**0.96**	0.95	90.6%
Prejudice	1.00	1.00	**1.00**	1.00	0.6%
Punctuation	0.94	0.95	**0.96**	0.95	100.0%
SwapNum	1.00	1.00	**1.00**	1.00	11.8%
VerbTense	0.98	0.97	**0.99**	0.97	73.4%
Twitter	0.93	0.93	**0.97**	0.95	100.0%
WordCase	0.00	**1.00**	**1.00**	0.22	100.0%

Table 17. Cohen's kappa scores for each model and other types of perturbations for the Hate Speech and Offensive Language task. The best score for each perturbation is highlighted

Natural Language Inference

The results for character level perturbation in this task fall between Grammatical Coherence and Hate Speech and Offensive Language in terms of kappa score:

Natural Language Inference (MNLI Corpus) - Character level						
Perturbation	% of characters per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Keyboard	1%	0.63	0.74	0.73	**0.74**	99.9%
	5%	0.45	0.53	0.54	**0.57**	99.9%
	10%	0.18	0.22	0.29	**0.33**	99.9%
Ocr	1%	0.63	0.73	0.72	**0.73**	99.8%
	5%	0.44	0.52	0.53	**0.55**	99.8%
	10%	0.21	0.24	0.32	**0.34**	99.8%
SpellingError	1%	0.68	**0.77**	0.75	0.74	98.7%
	5%	0.49	0.59	**0.60**	0.60	98.8%
	10%	0.28	0.36	**0.40**	0.39	98.9%
Typos	1%	0.65	**0.75**	0.72	0.73	93.3%
	5%	0.46	0.55	0.54	**0.56**	98.2%
	10%	0.22	0.27	0.33	**0.35**	99.6%

Table 18. Cohen's kappa scores for each model and character level perturbation type for the Natural Language Inference task. The best score for each perturbation is highlighted

The scores are between 0.3 and 0.75, indicating substantial agreement, especially when the inputs are perturbed only in 1%. Similar to HSOL, XLNet is more resilient to perturbed inputs. The drop in kappa score for DistilBERT and ELECTRA is greater than in Funnel Transformer.

Similar to the previous tasks, the type of relationship between the percentage of perturbed characters and Cohen's kappa is not clear, although it seems linear like in Grammatical Coherence.

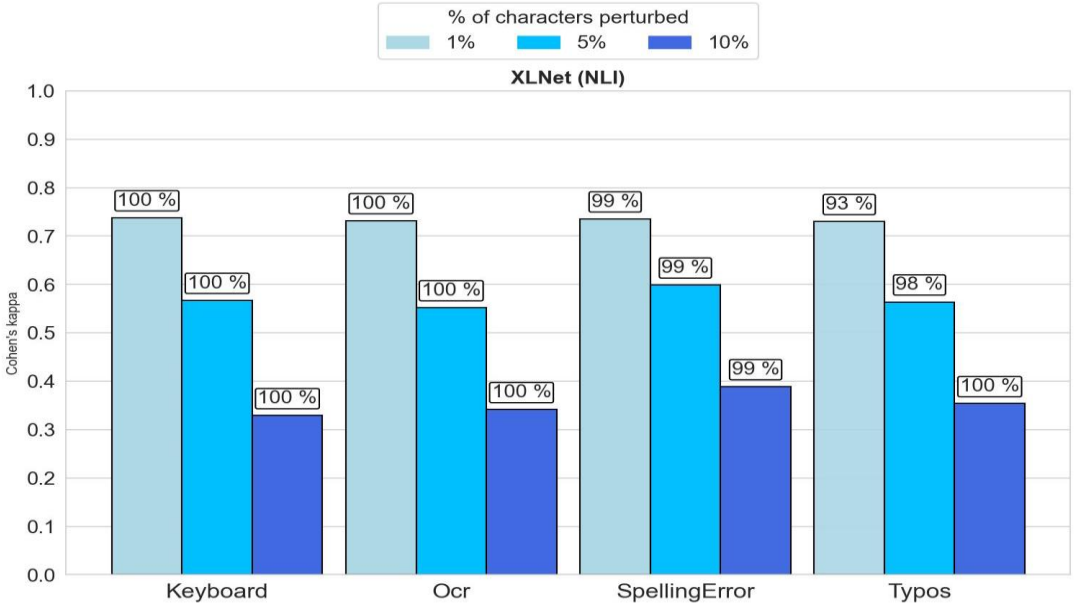

Figure 11. Cohen's kappa coefficient for character level perturbations in the finetuned XLNet model on the MNLI Corpus dataset. The value above the bars indicates the percentage of perturbed samples in the dataset

The results for word level perturbations are the best ones so far, with values around 0.6 and 0.75. DistilBERT is the most fragile model, with ELECTRA being superior to Funnel Transformer. XLNet is at the same level as ELECTRA.

Natural Language Inference (MNLI Corpus) - Word level						
Perturbation	% of words per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Swap Synonym Word Embedding	10%	**0.91**	0.87	0.86	0.87	19.3%
	20%	**0.91**	0.87	0.85	0.87	19.3%
	30%	**0.91**	0.87	0.85	0.87	19.3%
Swap Synonym WordNet	10%	0.70	**0.76**	0.74	0.75	99.3%
	20%	0.62	**0.70**	0.69	0.69	99.3%
	30%	0.54	**0.63**	0.61	0.63	99.3%

Table 19. Cohen's kappa scores for each model and word level perturbation type for the Natural Language Inference task. The best score for each perturbation is highlighted

The same observations can be made for other types of perturbations. XLNet and Funnel Transformer obtain similar kappas in all perturbations except Twitter, where

XLNet is superior. In the case of WordCase, the scores are better than in Grammatical Coherence and HSOL.

Natural Language Inference (MNLI Corpus) - Other					
Perturbation	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Contraction	0.95	**0.98**	0.97	0.98	11.0%
InsertAdv	0.63	**0.64**	0.62	0.63	94.7%
Prejudice	**0.99**	0.99	0.99	0.99	0.5%
Punctuation	0.83	**0.92**	0.90	0.90	100.0%
SwapNum	**0.97**	0.97	0.97	0.97	3.8%
VerbTense	0.90	**0.94**	0.92	0.92	68.7%
Twitter	0.56	**0.84**	0.73	0.81	100.0%
WordCase	0.35	**1.00**	**1.00**	0.49	99.9%

Table 20. Cohen's kappa scores for each model and other types of perturbations for the Natural Language Inference task. The best score for each perturbation is highlighted

Sentiment Analysis

Sentiment Analysis (Stanford Sentiment Treebank) - Character level						
Perturbation	% of characters per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Keyboard	1%	0.88	0.91	**0.93**	0.91	100.0%
	5%	0.69	0.75	**0.77**	0.77	100.0%
	10%	0.28	0.43	0.37	**0.47**	100.0%
Ocr	1%	0.86	**0.93**	0.93	0.92	100.0%
	5%	0.67	**0.77**	0.75	0.74	100.0%
	10%	0.28	0.45	0.34	**0.47**	100.0%
SpellingError	1%	0.91	0.94	**0.94**	0.92	99.9%
	5%	0.70	0.80	**0.81**	0.78	99.9%
	10%	0.52	0.62	0.63	**0.64**	99.9%
Typos	1%	0.90	0.92	**0.93**	0.89	99.9%
	5%	0.69	**0.76**	0.75	**0.76**	100.0%
	10%	0.32	0.45	0.32	**0.46**	99.9%

Table 21. Cohen's kappa scores for each model and character level perturbation type for the Sentiment Analysis task. The best score for each perturbation is highlighted

According to Table 21, Sentiment Analysis yields similar results to Hate Speech and Offensive Language. There is a clear distinction, though, and that is the fact that there is no clear dominant model: except DistilBERT, the other three models have a higher score

in certain perturbations, and it is not even consistent, since the best model for a perturbation varies depending on the percentage of perturbed characters.

However, XLNet seems to be the best model when the degree of perturbation is high, since it is the best model in all four character perturbations with a 10% of perturbed characters. DistilBERT is in the other end, keeping similar scores when the degree of perturbation is low, but failing to keep up when it is high. Figure 6 shows this, and it also shows that the previously mentioned linear relationship is not that clear in this task.

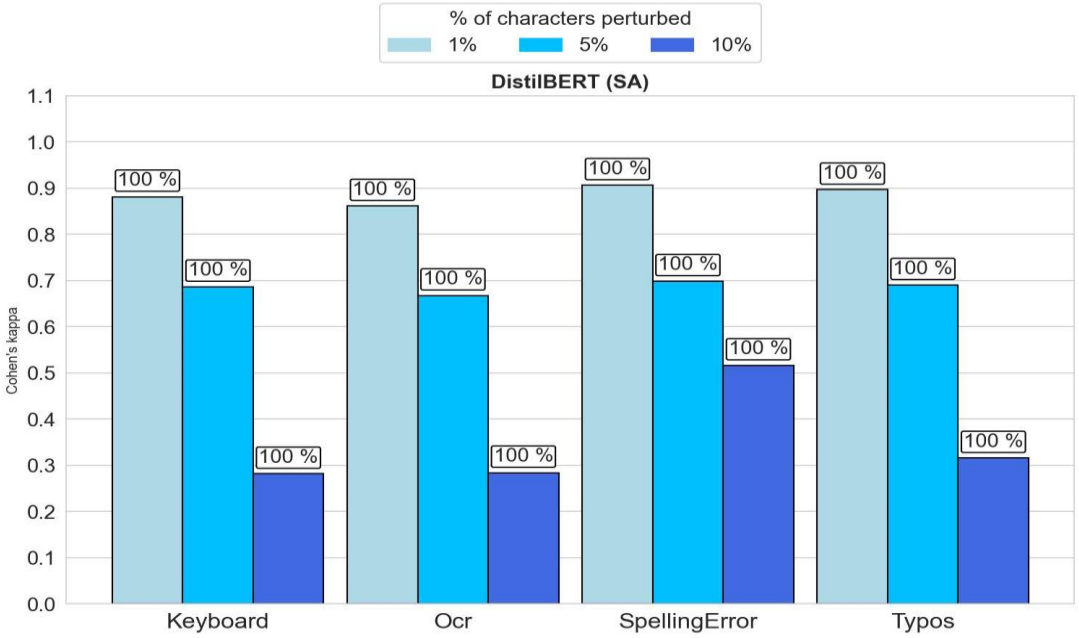

Figure 12. Cohen's kappa coefficient for character level perturbations in the finetuned DistilBERT model on the SST2 dataset. The value above the bars indicates the percentage of perturbed samples in the dataset.

The results for word level perturbations are different, though: Funnel Transformer is much more robust than XLNet, independently of the percentage of perturbed words.

Sentiment Analysis (Stanford Sentiment Treebank) - Word level						
Perturbation	% of words per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Swap Synonym Word Embedding	10%	1.00	1.00	**1.00**	1.00	7.4%
	20%	1.00	0.99	**1.00**	1.00	7.4%
	30%	1.00	1.00	**1.00**	1.00	7.4%
Swap Synonym WordNet	10%	0.88	0.92	**0.93**	0.91	99.4%
	20%	0.81	0.88	**0.89**	0.86	99.4%
	30%	0.73	0.82	**0.83**	0.79	99.4%

Table 22. Cohen's kappa scores for each model and word level perturbation type for the Sentiment Analysis task. The best score for each perturbation is highlighted

This is also the task where the proportion of perturbed samples in the dataset altered with the Swap Synonym Word Embedding perturbation is the least, indicating that this perturbation was not successful for the analysis: for instance, in this task the kappa coefficient for all the models is 1 in that perturbation, indicating perfect agreement, which is easy to achieve as barely any samples are being perturbed.

Regarding other perturbations, ELECTRA is again the model with the best scores, although Funnel Transformer yields similar scores. Once again, DistilBERT is the most sensitive model, but overall, except for the WordCase perturbation, the models are capable of outputting consistent predictions.

Sentiment Analysis (Stanford Sentiment Treebank) - Other					
Perturbation	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Contraction	0.98	**0.99**	0.99	0.98	27.0%
InsertAdv	0.88	**0.94**	0.93	0.89	90.7%
Prejudice	**1.00**	**1.00**	**1.00**	**1.00**	0.0%
Punctuation	0.94	0.97	**0.97**	0.95	100.0%
SwapNum	**1.00**	1.00	**1.00**	**1.00**	4.1%
VerbTense	0.94	**0.98**	0.96	0.95	71.3%
Twitter	0.93	**0.96**	0.96	0.94	100.0%
WordCase	0.00	**1.00**	**1.00**	0.35	100.0%

Table 23. Cohen's kappa scores for each model and other types of perturbations for Sentiment Analysis task. The best score for each perturbation is highlighted

Semantic Similarity

Semantic Similarity (PAWS) - Character level						
Perturbation	% of characters per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Keyboard	1%	0.66	0.70	0.75	**0.85**	100.0%
	5%	0.13	0.08	0.22	**0.37**	100.0%
	10%	0.04	0.03	0.10	**0.17**	100.0%
Ocr	1%	0.69	0.65	0.77	**0.84**	100.0%
	5%	0.21	0.08	0.28	**0.38**	100.0%
	10%	0.14	0.09	0.30	**0.31**	100.0%
SpellingError	1%	0.73	0.77	0.82	**0.87**	100.0%
	5%	0.30	0.30	0.43	**0.55**	100.0%
	10%	0.24	0.29	0.42	**0.51**	100.0%
Typos	1%	0.67	0.70	0.79	**0.85**	100.0%
	5%	0.16	0.12	0.26	**0.44**	100.0%
	10%	0.07	0.06	0.20	**0.32**	100.0%

Table 24. Cohen's kappa scores for each model and character level perturbation type for the Semantic Similarity task. The best score for each perturbation is highlighted

Semantic Similarity yields similar results as Natural Language Inference in terms of Cohen's kappa coefficient. However, the drop from a perturbation percentage of 1% to 5% is much greater than from 5% to 10%, contrary to all previous tasks. This is shown in Figure 7.

XLNet is far superior than the rest of the models, surpassing Funnel Transformer (the second best model) sometimes by 0.1 or even almost 0.2. ELECTRA is more robust than DistilBERT when the degree of perturbation is low, but it quickly falls to more perturbed inputs.

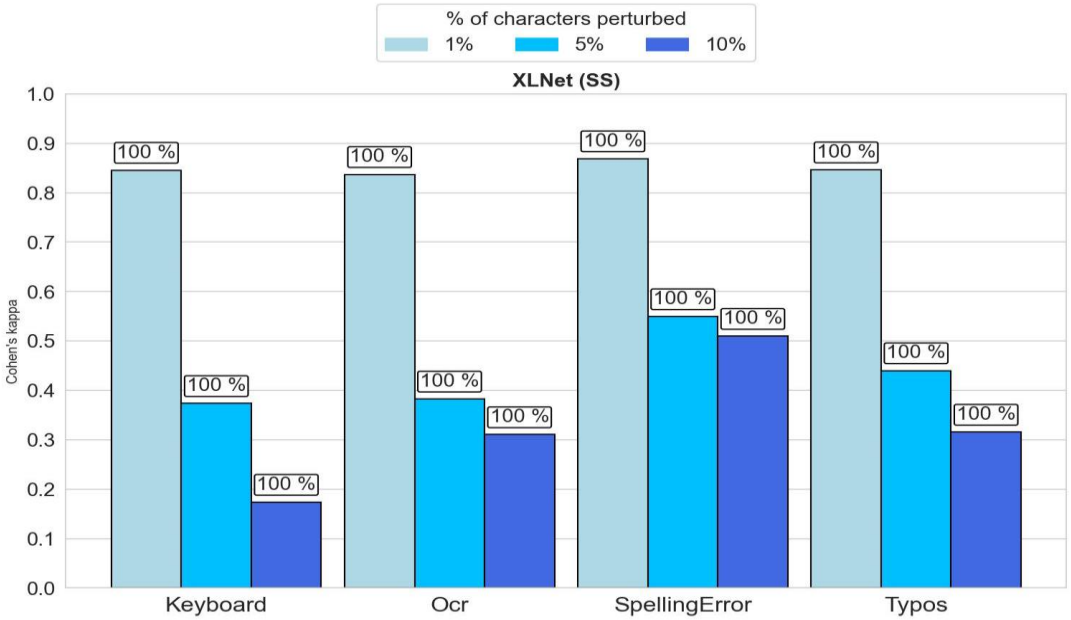

Figure 13. Cohen's kappa coefficient for character level perturbations in the finetuned DistilBERT model on the PAWS dataset. The value above the bars indicates the percentage of perturbed samples in the dataset. Opposite to previous tasks, the bigger drop in kappa score is from 1% to 5% perturbation percentage

When talking about word level perturbations, PAWS is the only dataset where the perturbation Swap Synonym Word Embedding has a significant proportion of perturbed samples, and therefore greatly affects the level of agreement between the predictions of the normal dataset and the perturbed dataset.

Semantic Similarity (PAWS) - Word level						
Perturbation	% of words per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Swap Synonym Word Embedding	10%	0.46	0.37	0.44	**0.50**	72.1%
	20%	0.46	0.37	0.44	**0.50**	72.1%
	30%	0.46	0.38	0.45	**0.50**	72.1%
Swap Synonym WordNet	10%	0.76	0.84	0.86	**0.88**	100.0%
	20%	0.62	0.72	0.76	**0.80**	100.0%
	30%	0.59	0.71	0.74	**0.78**	100.0%

Table 25. Cohen's kappa scores for each model and word level perturbation type for the Semantic Similarity task. The best score for each perturbation is highlighted

Once again, XLNet obtains better scores, although in Swap Synonym Word Embedding there is no difference in the percentage of perturbed words per sample. The same tendency as with character level perturbations appears in Swap Synonym

WordNet, although the overall drop in kappa is not that high, with the biggest difference being from 0.76 to 0.59 (-0.15) from DistilBERT.

The XLNet dominance breaks when analysing other perturbations, since ELECTRA and Funnel Transformer obtain better scores in Punctuation, VerbTense and Twitter, as shown in Table 26. The WordCase perturbation seems to affect DistilBERT much more detrimentally than XLNet, which gets a Cohen's kappa of 0.64, the highest value for the model in any dataset perturbed with this type of perturbation. It looks like Semantic Similarity deviates from the common patterns the previous NLP tasks showed.

Semantic Similarity (PAWS) - Other					
Perturbation	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Contraction	0.99	1.00	**1.00**	1.00	5.01%
InsertAdv	0.75	0.49	0.64	**0.77**	99.75%
Prejudice	0.98	0.96	0.98	**0.98**	2.86%
Punctuation	0.87	**0.96**	0.96	0.95	100.00%
SwapNum	0.96	0.75	0.98	**0.98**	40.15%
VerbTense	0.92	**0.98**	0.97	0.96	76.41%
Twitter	0.56	0.87	**0.95**	0.94	100.00%
WordCase	0.13	**1.00**	**1.00**	0.64	100.00%

Table 26. Cohen's kappa scores for each model and other types of perturbations for the Semantic Similarity task. The best score for each perturbation is highlighted

In summary, the different tasks provide different levels of agreement, with Grammatical Coherence being the most disruptive one, yielding the lowest Cohen's kappa score of all five datasets, and Sentiment Analysis being the most consistent one with high kappa values near to 0.9.

XLNet is generally the most robust model, especially in character level perturbations and challenging datasets like PAWS. However, ELECTRA and Funnel Transformer also present high values when the datasets are perturbed with other types of perturbations, such as TwitterType or Punctuation. DistilBERT seems to be more sensitive to perturbed inputs, particularly when the percentage of perturbed characters or words is high.

4.2.3. Aggregated analysis

To provide a comprehensive analysis, this section presents aggregated calculations that answer the three main objectives of this project, aligning with the three dimensions studied: models, tasks, and perturbations. It is important to note that the score presented in the following tables takes into account the proportion of perturbed samples in the datasets. This approach prevents datasets with minimal perturbations, such as the Contraction or Prejudice perturbations, from inflating the final scores.

Also, the WordCase perturbation was not employed to compute these totals, since, as explained previously, DistilBERT and XLNet are highly affected by it due to their tokenisation processes.

The group of tables 27 shows the average kappa score per model, per NLP task and per type of perturbation.

Model	Score
XLNet	0.487
Funnel Transformer	0.472
ELECTRA	0.464
DistilBERT	0.445

NLP Task	Score
SA	0.564
HSOL	0.536
SS	0.487
NLI	0.451
GC	0.297

Perturbation type	Score
CHARACTER	0.495
OTHER	0.468
WORD	0.438

Table 27. From left to right, average kappa score per model, per NLP task and per perturbation

In concordance with the detailed analysis performed, the results indicate that XLNet is the most robust of the four models evaluated, with an average kappa score of 0.487. Funnel Transformer and ELECTRA follow behind close with 0.472 and 0.464, respectively. Finally, DistilBERT is more sensitive to perturbed inputs, having a score of 0.445.

Regarding NLP tasks, Grammatical Coherence is by far the most challenging task to the models, with a low 0.297, indicating a very low level of agreement. In contrast, Sentiment Analysis and Hate Speech and Offensive Language have less impact on the models, with scores of 0.536 and 0.564 respectively. Meanwhile, Semantic Similarity and Natural Language Inference land in the middle with averages of 0.487 and 0.451.

Finally, the results suggest that word-level perturbations are more detrimental to language models than character-level or other types of perturbations, although the difference in average kappa is not significant. The group of tables 28 shows the average

kappa values for character-level, word-level and other perturbations respectively. Perturbations in red indicate that the proportion of perturbed samples in the dataset is less than 50%.

Chracter perturbation type	Score
SpellingError	0.535
Ocr	0.488
Typos	0.483
Keyboard	0.474

Word perturbation type	Score
SSWN	0.632
SNE	0.243

Other perturbation type	Score
Punctuation	0.898
Twitter	0.836
InsertAdv	0.642
WordCase	0.632
VerbTense	0.608
Contraction	0.169
SwapNum	0.115
Prejudice	0.009

Table 28. From left to right, average kappa score for character level, word level and other perturbations. Values in red indicate that the mean proportion of perturbed samples in the dataset is less than 50%

However, it is worth noting that character-level perturbations were the most challenging for the models when analysed in isolation and not as a group of perturbations. The models produced more consistent outputs with inputs perturbed by punctuation and Twitter. This finding seems to contradict the previous statement that word-level perturbations have the greatest impact. Upon further investigation, this discrepancy could be attributed to the fact that word-level and other types of perturbations have lower scores due to less common perturbations, such as Contraction, Prejudice and SwapNum for other perturbations, and Swap Synonym Word Embeddings for word-level perturbations. Consequently, these anomalies lowered the average kappa values for the corresponding perturbation types.

To address this issue, we removed unrepresentative perturbations and recalculated the average kappa scores per perturbation type. The results are shown in Table 29, which changes which type of perturbation has the most significant effect.

Perturbation type	Score
OTHER	0.723
WORD	0.632
CHARACTER	0.495

Table 29. Average kappa score per type of perturbation when non-represented perturbations are removed

In summary, the results suggest that XLNet is the most robust model, while Grammatical Coherence is the most challenging NLP task. Furthermore, while word-

level perturbations are generally more detrimental to language models, character-level perturbations may be more difficult to analyse in isolation. Therefore, researchers and developers should carefully consider the type of perturbation applied in their evaluation strategy in order to obtain accurate evaluation results.

5. Conclusions

This works proves that Large Language Models, although powerful and able to remain consistent in their predictions in some specific circumstances (like with some perturbations such as adding punctuation marks, links or usernames), can still become volatile when the input data is noisy or altered by some kind of natural perturbation.

This volatility is not the same in all models: XLNet is the more resilient one, being able to achieve Cohen's kappa values of 0.15 at worst and 0.9 at best. Funnel Transformer and ELECTRA follow close, and the most affected model is DistilBERT.

One aspect to consider is that XLNet takes much more time to finetune than the other models, so the advantage in robustness comes at the cost of time and energy resources. In specific scenarios or concrete NLP tasks, a smaller model with almost similar results like ELECTRA or Funnel Transformer could be a better choice.

Regarding NLP tasks, Grammatical Coherence is the most brittle one, achieving average Cohen's kappa values of 0.29. No model is able to achieve a score of 0.50 in the CoLA corpus, especially with character level perturbations. This could be explained by assuming that current LLMs overfit for this task, not fully understanding the grammatical rules of a language and wildly changing their output if an insignificant change like a typo occurs.

On the other hand, tasks with much broader goals such as the identification of emotions (Sentiment Analysis) and the detection of strong negative emotions (Hate Speech and Offensive Language) seem less affected by perturbations. Models finetuned for these tasks understand the key components that help make a prediction, and therefore are less volatile.

Tasks involving a logical reasoning of the language such as Semantic Similarity and Natural Language Inference land somewhere in the middle, with values close to 0.48 in some cases.

Perturbation-wise, and in agreement with (Moradi & Samwald, 2021) some character level perturbations are more detrimental to the models' robustness. The most natural ones such as common spelling mistakes can lead to kappa scores of 0.4 and 0.5, even when the sentence is highly perturbed (10% of characters are misspelled).

Miscellaneous perturbations vary in success: adding punctuation marks and links and usernames similar to those found on Twitter do not have that much of an impact, with models reaching high scores of 0.8 and 0.9. In contrast, the insertion of adverbs before verbs and the capitalisation of letters makes the models less robust.

In conclusion, the different characteristics of each model provide a series of strengths and weaknesses that help or detriment their robustness against different types of perturbations: in some cases, they are able to maintain great levels of consistency in their predictions, but most times Large Language Models succumb to reality of non-curated, real world natural data.

6. Legacy and relationship with studies

This work has studied the robustness of modern Large Language Models in common NLP tasks. As such, the findings can help model designers create architectures that mitigate the impact of these perturbations.

All the results about the finetuning and the evaluation phase, as well as the code to produce them can be found in this repository. The perturbed datasets are publicly available and uploaded to HuggingFace in the username DaniFrame. The rest of materials used in this report (such as tables, figures and additional information) are provided in Annex I, II and III: Annex I is about the relationship of this project with the Sustainable Development Goals (SDG); Annex II has additional information regarding the datasets and the models; and Annex III contains all figures and tables regarding the finetuning and evaluation phase.

Regarding the relationship between this work and the studies in the Degree of Data Science, the main connection is clearly the discipline of Natural Language Processing, which is one of the main subjects in the third year of the degree.

There are other parts of the studies that have significantly helped with the conception and execution of this project: from the basics of programming in Python to the use of Machine Learning and Deep Leaning libraries like transformers and PyTorch. There is also a high degree of project methodology, organisation and planning achieved thanks to the subjects of Project I, II and III. Overall, this project really relies on a varied set of skills developed during the studies.

Furthermore, soft skills have also been necessary to successfully complete this project. Even though practically all of them were applied to an extent, those that have been put into practice to a higher extent were the following:

- CT 01 – Understanding and integration: the integration of two different technologies (training of LLMs and creation of perturbations) through libraries such as transformers, PyTorch and textflint is the heart of the implementation of this project.

- CT 02 – Application and practical thinking: this work heavily relies on empirical results derived from experiments, and as such the theoretical knowledge obtained in the degree has been applied to various components such as the finetuning of models or making sure the perturbations were being applied correctly.

- CT 11 – Continuous learning: this project has demanded great adaptation capabilities and the ability to continuously gain more knowledge: libraries such as transformers and textflint were new to me and had to be learnt. Moreover, the initial implementation of the finetuning phase was made using tensorflow, another Deep Learning framework, but unsurpassable errors led to the change to PyTorch.

7. Future work and improvements

Even though this word sheds light on the robustness of modern LLMs, there is still much more room for improvement considering the limitations of time and resources imposed by the nature of this Final Year Project.

One aspect to upgrade would be the finetuning phase: due to time constraints, only two hyperparameters were optimised. Other options such as the batch size, different regulations techniques such as L1 (Ranstam & Cook, 2018) and L2 (Hilt & Seegrist, 1977), the use of dropout, etc., given more time and better hardware, could have been optimised. The use of cross validation could have also been beneficial.

This idea of “more and better” is also applicable to the perturbations: textflint has 20 general transformations (a subset of them were used in this project), but also has 60 task-specific transformations, including perturbations for Sentiment Analysis and Natural Language Inference, that could be used to complete the analysis of this project.

Futures lines of development may also go in the direction of implementing this type of perturbation robustness into the evaluation of new models. It would also be interesting to analyse very powerful models with and without finetuning, such as the GPT family, Bard or Palm, as they become available.

All this could be facilitated in two non-exclusive ways:

- Create a database of perturbed datasets, where different curated datasets and their altered counterparts could be stored to be used as benchmarks for different tasks. My repository at Hugging Face is a starting point.

- Upon creating a new model, create (or obtain from the database) a perturbed dataset and evaluate the change in predictions or the model, yielding a similar analysis as the one made in this work, but custom-made for that specific model.

Finally, it would also be interesting to finetune models with perturbed datasets, to determine if their robustness increases and analyse the tradeoff between performance and robustness.

8. Acknowledgments

I would like to thank Jose Hernandez Orallo and Fernando Martínez Plumned for their tutorship and guidance throughout the project, the final version of this work would have been impossible without the.

I would also like to acknowledge the UPV Data Mining, Machine Intelligence and Inductive Programming (DMIP) team, who gave me access to the computational resources needed to accomplish this task.

9. Bibliography

Adel, H., Dahou, A., Mabrouk, A., Elsayed Abd Elaziz, M., Kayed, M., El-henawy, I., . . . Ali, A. (2022, 01). Improving Crisis Events Detection Using DistilBERT with Hunger Games Search Algorithm. *Mathematics, 10*, 447. doi:10.3390/math10030447

Belinkov, Y., & Bisk, Y. (2018). Synthetic and Natural Noise Both Break Neural Machine Translation.

Bengio, Y., Ducharme, R., & Vincent, P. (2000). A Neural Probabilistic Language Model. (T. Leen, T. Dietterich, & V. Tresp, Eds.) *Advances in neural information processing systems, 13*. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf

Brei, V. (2020, 1). Machine Learning in Marketing: Overview, Learning and Strategies, Applications and Future Developments. *Foundations and Trends ® in Marketing, 14*(3), 173-236. doi:10.1561/1700000065

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., . . . others. (2020). Language Models are Few-Shot Learners. *Advances in neural information processing systems*, 1877-1901.

Burnell, R., Schellaert, W., Burden, J., Ullman, T. D., Martinez-Plumed, F., Tenenbaum, J. B., . . . others. (2023). Rethink reporting of evaluation results in AI. *Science, 380*(6641), 136-138.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.

Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training Text Enoders as Discriminators rather than Generators. In *ICLR.* Retrieved from https://openreview.net/pdf?id=r1xMH1BtvB}

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement, 20*(1), 37-46.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. *Proceedings of the international AAAI conference on web and social media, 11*(1), 512-515.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171-4186). Minneapolis, Minnesota: Association for Compuational Linguistics. doi:10.18653/v1/N19-1423

Fellbaum, C. (1998). WordNet: An Electronic Lexical Database.

Hilt, D. E., & Seegrist, D. W. (1977). Ridge, a computer program for calculating ridge regression estimates. *236*, 10. Retrieved from https://www.biodiversitylibrary.org/item/137258

Hochreiter, S., & Schmidhuber, J. (1997, 12). Long Short-term Memory. *Neural Computation, 9*(8), 1735-1780. doi:10.1162/neco.1997.9.8.1735

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010, 1). Recurrent neural network based langauge model. *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010, 2*, 1045-1048.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems, 26*.

Moradi, M., & Samwald, M. (2021). Evaluating the Robustness of Neural Language Models to Input Perturbations. Retrieved 3 2023

OpenAI. (2022, 11 30). *Introducing ChatGPT*. Retrieved from https://openai.com/blog/chatgpt

OpenAI. (2023). GPT-4 Technical Report.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., . . . Lerer, A. (2017). Automatic differentiation in PyTorch.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532-1543.

Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine, 3*(1), 4-16. doi:10.1109/MASSP.1986.1165342

Ranstam, J., & Cook, J. A. (2018, 8). LASSO regression. *British Journal of Surgery, 105*(10), 1348-1348. doi:10.1002/bjs.10895

Reshamwala, A., Mishra, D. P., & Prajakta. (2013, 2). Review on Natural Languge Processing. *IRACST - Engineering Science and Technology: An International Journal (ESTIJ), 3*(1), 113-116.

Ribeiro, M. T., Wu, T., Guestrin, C., & Singh, S. (2020). Beyond Accuracy: Behavioral Testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4092-4912). Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.442

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013, 10). Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631-1642. Retrieved from https://aclanthology.org/D13-1170

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *Advances in neural information processing systems, 27*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. *Advances in neural information processing systems, 30*.

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., & others. (2018). Deep learning for computer vision: A brief review. *Computation intelligence and neuroscience, 2018*.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.

Wang, X., Liu, Q., Gui, T., Zhang, Q., Zou, Y., Zhou, X., . . . others. (2021). Textflint: Unified multilingual robustness evaluation toolkit for natural language processing. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, 347-355.

Warstadt, A., Singh, A., & Bowman, S. R. (2018). Neural Network Acceptability Judgments. *arXiv preprint arXiv:1805.12471*.

Williams, A., Nangia, N., & Bowman, S. (2018). A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 1112-1122). New Orleans, Louisiana: Association for Computational Linguistics. Retrieved from http://aclweb.org/anthology/N18-1101

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . others. (2020). HuggingFace's Transformers: State-of-the-art Natural Language Processing.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Languge Understanding. *Advances in neural information processing systems, 32*.

Yu, K.-H., Beam, A. L., & Kohane, I. S. (2018). Artificial intelligence in healthcare. *Natura biomeical engineering, 2*(10), 719-731.

Zeng, G., Qi, F., Shou, Q., Zhang, T., Ma, Z., Hou, B., . . . Sun, M. (2021, 8). OpenAttack: An Open-source Textual Adversarial Attack Toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations* (pp. 363-371). Association for Computational Linguistics. doi:10.18653/v1/2021.acl-demo.43

Zhang, S., Yu, H., & Zhu, G. (2021, 10). An emotional Classification method of Chinese short comment text based on ELECTRA. *Connection Science, 34*, 1-20. doi:10.1080/09540091.2021.1985968

Zhang, Y., Baldridge, J., & He, L. (2019). PAWS: Paraphrase Adversaries from Word Scrambling. *arXiv preprint arXiv:1904.01130*.

Zihang, D., Guokun, L., Yiming, Y., & Quoc, V. L. (2020). Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing. *Advances in neural information processing systems*, 4271-4282.

ANNEX I. Sustainment Development Goals

Sustainable Development Goals	High	Medium	Low	Not applicable
SGD 1. No poverty				X
SDG 2. Zero hunger				X
SDG 3. Good health and well-being				X
SDG 4. Quality education				X
SDG 5. Gender equality			X	
SDG 6. Clean water and sanitation				X
SDG 7. Affordable and clean energy				X
SDG 8. Decent work and economic growth				X
SDG 9. Industry, Innovation and Infrastructure	X			
SDG 10. Reduce inequality			X	
SDG 11. Sustainable cities and communities				X
SDG 12. Responsible consumption and production				X
SDG 13. Climate action				X
SDG 14. Life below water				X
SDG 15. Life on land				X
SDG 16. Piece, justice and strong institutions			X	
SDG 17. Partnership for the goals				X

Table 1. Project relationship with the Sustainable Development Goals

Reflection on the relationship of the TFG with the SDGs and with the most related SDG(s):

This project is made as an End of Degree Project in the UPV Data Science Degree. As such, the most related SDG is definitely SDG 9: Industry, Innovation and Infrastructure.

SDG 9 aims to promote inclusive and sustainable industrialisation, foster innovation, and enhance infrastructure. In the context of the project, evaluating the robustness of NLP models helps advance technological innovation by identifying areas where these models can be strengthened and made more reliable. The findings can help in the design of more efficient and robust models. More powerful and exact tools are essential to the economic growth of countries, specially in this modern era of information and technology where institutions and companies aim to be more digitalised.

Moreover, unstructured data such as text is difficult to analyse and exploit, so this work gives a general direction on how to improve the way we look at this kind of data.

The project also has connections to several other SDGs. It relates to SDG 5: Gender Equality, as it contributes to the understanding of potential biases and discriminatory patterns that might exist in NLP models. This is evaluated via a perturbation called “Prejudice”, that changes masculine names to feminine ones and vice versa.

NLP models often learn from existing language data, and even though most datasets are curated and pre-analysed before being used for training, societal biases can still exist. By evaluating the robustness of these models, researchers can uncover and address biases related to gender. This research can ultimately contribute to promoting gender equality, inclusivity, and non-discrimination in the use of AI technologies. For this exact reason, this work is also related SDG 10: Reduced Inequalities.

Finally, this project is also relevant to SDG 16: Peace, Justice, and Strong Institutions. By identifying vulnerabilities and potential biases in these models, the research helps build stronger institutions that make fair and unbiased decisions. NLP models are increasingly used in various domains, including legal systems, governance, and access to justice. Ensuring that these models are robust and reliable is crucial for maintaining transparency, accountability, and fairness in decision-making processes. Furthermore, there are concrete cases where the use of large language models can be

used in underdeveloped countries, where LLMs are not trained with their original languages due to the lack of data. If the population has basic knowledge of English, and therefore are prone to make mistakes, these models could still assist the population without losing its exactness, becoming another tool to mitigate the impact of inequalities and promote piece.

In summary, while primarily related to SDG 9: Industry, Innovation, and Infrastructure, this End of Degree Project also has significant connections to SDGs 5, 10, and 16. By addressing gender equality, reduced inequalities, and the promotion of peace, justice, and strong institutions, the research contributes to a more inclusive and sustainable development. It highlights the importance of improving the reliability and fairness of AI technologies, while also fostering innovation and ensuring their responsible use for the benefit of all individuals and communities.

ANNEX II. Models and datasets

LIST OF TABLES

Table 1. DistilBERT for sequence classification parameters _____ *1*
Table 2. ELECTRA for sequence classification parameters _____ *2*
Table 3. Funnel Transformer for sequence classification parameters _____ *3*
*Table 4. XLNet for sequence classification parameters*_____ *4*

Models

DistilBERT

Parameter	Value
_name_or_path	**distilbert-base-cased**
activation	gelu
architectures	DistilBertForSequenceClassification
attention_dropout	0.1
dim	768
dropout	0.1
hidden_dim	3072
initializer_range	0.02
max_possition_embeddings	512
model_type	distilbert
n_heads	12
n_layers	6
output_past	true
pad_toke_id	0
problem_type	single_label_classification
qa_dropout	0.1
seq_classif_dropout	0.2
sinusoidal_pos_embs	false
tie_wieights	true
torch_dtype	float32
transformers_version	4.29.2
vocab_size	28996

Table 1. DistilBERT for sequence classification parameters

ELECTRA

Parameter	Value
_name_or_path	**google/electra-base-discriminator**
architectures	ELECTRAForSequenceClassification
attention_probs_dropout_prob	0.1
classifier_dropout	NULL
embedding_size	768
hidden_act	gelu
hidden_dropout_prob	0.1
hidden_size	768
initializer_range	0.02
intermediate_size	3072
layer_norm_eps	1.00E-12
max_position_embeddings	512
model_type	electra
num_attention_heads	12
num_hidden_layers	12
pad_token_id	0
position_embedding_type	absolute
problem_type	single_label_classification
summary_activation	gelu
summary_last_dropout	0.1
summary_type	first
summary_use_proj	true
torch_dtype	float32
transformers_version	4.29.2
type_vocab_size	2
use_cache	true
vocab_size	30522

Table 2. ELECTRA for sequence classification parameters

Funnel Transformer

Parameter	Value
_name_or_path	**funnel-transformer/small**
activation_dropout	0
architectures	FunnelForSequenceClassification
attention_dropout	0.1
attention_type	relative_shift
block_repeats	3 blocks of 1
block_sizes	3 blocks of size 4
d_head	64
d_inner	3072
d_model	768
hidden_act	gelu_new
hidden_dropout	0.1
initializer_range	0.1
initializer_std	NULL
layer_norm_esp	1.00E-09
max_position_embeddings	512
model_type	funnel
n_head	12
num_decoder_layers	2
pool_q_only	true
pooling_type	mean
problem_type	single_label_classification
rel_attn_type	factorised
separate_cls	true
torch_dtype	float32
transformers_version	4.29.2
truncate_seq	true
type_vocab_size	3
vocab_size	30522

Table 3. Funnel Transformer for sequence classification parameters

XLNet

Parameter	Value
_name_or_path	**xlnet-base-casedd**
architectures	XLNetForSequenceClassification
attn_type	bi
bi_data	false
bos_token_id	1
clamp_len	-1
d_head	64
d_inner	3072
d_model	768
dropout	0.1
end_n_top	5.00E+00
eos_token_id	2
ff_activation	gelu
initializer_range	0.02
layer_norm_eps	1.00E-12
model_type	xlnet
mem_len	NULL
n_head	12
n_layer	12
pad_token_id	5
problem_type	single_label_classification
reuse_len	NULL
same_length	false
start_n_top	5
summary_activation	tanh
summary_last_dropout	0.1
summary_type	last
summary_use_proj	true
torch_dtype	float32
transformers_version	4.29.2
untie_r	true
use_mems_eval	true
use_mem_train	false
vocab_size	32000

Table 4. XLNet for sequence classification parameters

Datasets

CoLA

Nº of samples per split:

- *Train: 8851*
- *Validation: 1043*
- *Test: 1063*

Class distribution per split:

- *Train: 30% **Class 0**, 70 % **Class 1***
- *Validation: 30% **Class 0**, 70 % **Class 1***
- *Test: labels unknown*

Hate speech offensive

Nº of samples per split:

- *Train: 24783 -> Needs train-val-test partition*

Class distribution per split:

- *Train: 6% **Class 0**, 77% **Class 1**, 17% **Class 2***

MNLI

Nº of samples per split

- *Train: 392702 (subsampling needed)*
- *Validation_matched: 9815*
- *Test_matched: 9796*

Class distribution per split

- *Train: 33% **Class 0**, 33% **Class 1**, 33% **Class 2***
- *Validation_matched: 35% **Class 0**, 32% **Class 1**, 33% **Class 2***
- *Test_matched: labels unknown*

SST2

Nº of samples per split

- *Train: 67349 (subsampling needed)*
- *Validation: 827*
- *Test: 1281*

Class distribution per split

- *Train: 44% **Class 0**, 56% **Class 1***
- *Validation: 49% **Class 0**, 51% **Class 1***
- *Test: labels unknown*

PAWS

Nº of samples per split:

- *Train: 49401 (subsampling needed)*
- *Validation: 8000*
- *Test: 8000*

Class distribution per split:

- *Train: 56% **Class 0**, 44% **Class 1***
- *Validation: 56 % **Class 0**, 44% **Class 1***
- *Test: 56 % **Class 0**, 44% **Class 1***

ANNEX III. Figures and tables

Contents

Figures 4

Example of distribution shift 4

Attention matrix 4

Models 5

Evaluation 7

Character level perturbations 7

Word level perturbations 17

Other perturbations 27

Tables 37

Methodology 37

Finetuning 40

Detailed evaluation 41

Character level perturbations 41

Word level perturbations 43

Other perturbations 45

Aggregated evaluation 47

LIST OF FIGURES

Figure 1. Example of distribution shifts ____ *4*

Figure 2. Visual representation of attention and self-attention ____ *4*

Figure 3. The DistilBERT model and its components ____ *5*

Figure 4. The ELECTRA model and its components ____ *5*

Figure 5. The XLNet model and its components ____ *6*

Figure 6. The funnel transformer model and its components ____ *6*

Figure 7. Confusion matrix to explain the computation of the Kappa Cohen's score ____ *6*

Figure 8. Cohen's kappa evaluation for character level perturbations of DistilBERT in GC ____ *7*

Figure 9. Cohen's kappa evaluation for character level perturbations of ELECTRA in GC ____ *7*

Figure 10. Cohen's kappa evaluation for character level perturbations of Funnel Transformer in GC ____ *8*

Figure 11. Cohen's kappa evaluation for character level perturbations of XLNet in GC ____ *8*

Figure 12. Cohen's kappa evaluation for character level perturbations of DistilBERT in HSOL ____ *9*

Figure 13. Cohen's kappa evaluation for character level perturbations of ELECTRA in HSOL ____ *9*

Figure 14. Cohen's kappa evaluation for character level perturbations of Funnel Transformer in HSOL _ *10*

Figure 15. Cohen's kappa evaluation for character level perturbations of XLNet in HSOL ____ *10*

Figure 16. Cohen's kappa evaluation for character level perturbations of DistilBERT in NLI ____ *11*

Figure 17. Cohen's kappa evaluation for character level perturbations of ELECTRA in NLI ____ *11*

Figure 18. Cohen's kappa evaluation for character level perturbations of Funnel Transformer in NLI ___ *12*

Figure 19. Cohen's kappa evaluation for character level perturbations of XLNet in NLI 12
Figure 20. Cohen's kappa evaluation for character level perturbations of DistilBERT in SA 13
Figure 21. Cohen's kappa evaluation for character level perturbations of ELECTRA in SA 13
Figure 22. Cohen's kappa evaluation for character level perturbations of Funnel Transformer in SA 14
Figure 23. Cohen's kappa evaluation for character level perturbations of XLNet in SA 14
Figure 24. Cohen's kappa evaluation for character level perturbations of DistilBERT in SS 15
Figure 25. Cohen's kappa evaluation for character level perturbations of ELECTRA in SS 15
Figure 26. Cohen's kappa evaluation for character level perturbations of Funnel Transformer in SS n 16
Figure 27. Cohen's kappa evaluation for character level perturbations of XLNet in SS 16
Figure 28. Cohen's kappa evaluation for word level perturbations of DistilBERT in GC 17
Figure 29. Cohen's kappa evaluation for word level perturbations of ELECTRA in GC 17
Figure 30. Cohen's kappa evaluation for word level perturbations of Funnel Transformer in GC 18
Figure 31. Cohen's kappa evaluation for word level perturbations of XLNet in GC 18
Figure 32. Cohen's kappa evaluation for word level perturbations of DistilBERT in HSOL 19
Figure 33. Cohen's kappa evaluation for word level perturbations of ELECTRA in HSOL 19
Figure 34. Cohen's kappa evaluation for word level perturbations of Funnel Transformer in HSOL 20
Figure 35. Cohen's kappa evaluation for word level perturbations of XLNet in HSOL 20
Figure 36. Cohen's kappa evaluation for word level perturbations of DistilBERT in NLI 21
Figure 37. Cohen's kappa evaluation for word level perturbations of ELECTRA in NLI 21
Figure 38. Cohen's kappa evaluation for word level perturbations of Funnel Transformer in NLI 22
Figure 39. Cohen's kappa evaluation for word level perturbations of XLNet in NLI 22
Figure 40. Cohen's kappa evaluation for word level perturbations of DistilBERT in SA 23
Figure 41. Cohen's kappa evaluation for word level perturbations of ELECTRA in SA 23
Figure 42. Cohen's kappa evaluation for word level perturbations of Funnel Transformer in SA 24
Figure 43. Cohen's kappa evaluation for word level perturbations of XLNet in SA 24
Figure 44. Cohen's kappa evaluation for word level perturbations of DistilBERT in SS 25
Figure 45. Cohen's kappa evaluation for word level perturbations of ELECTRA in SS 25
Figure 46. Cohen's kappa evaluation for word level perturbations of Funnel Transformer in SS 26
Figure 47. Cohen's kappa evaluation for word level perturbations of XLNet in SS 26
Figure 48. Cohen's kappa evaluation for other perturbations of DistilBERT in GC 27
Figure 49. Cohen's kappa evaluation for other perturbations of ELECTRA in GC 27
Figure 50. Cohen's kappa evaluation for other perturbations of Funnel Transformer in GC 28
Figure 51. Cohen's kappa evaluation for other perturbations of XLNet in GC 28
Figure 52. Cohen's kappa evaluation for other perturbations of DistilBERT in HSOL n 29
Figure 53. Cohen's kappa evaluation for other perturbations of ELECTRA in HSOL 29
Figure 54. Cohen's kappa evaluation for other perturbations of Funnel Transformer in HSOL 30
Figure 55. Cohen's kappa evaluation for other perturbations of XLNet in HSOL 30
Figure 56. Cohen's kappa evaluation for other perturbations of DistilBERT in NLI 31
Figure 57. Cohen's kappa evaluation for other perturbations of ELECTRA in NLI 31
Figure 58. Cohen's kappa evaluation for other perturbations of Funnel Transformer in NLI 32
Figure 59. Cohen's kappa evaluation for other perturbations of XLNet in NLI 32
Figure 60. Cohen's kappa evaluation for other perturbations of DistilBERT in SA 33
Figure 61. Cohen's kappa evaluation for other perturbations of ELECTRA in SA 33
Figure 62. Cohen's kappa evaluation for other perturbations of Funnel Transformer in SA 34
Figure 63. Cohen's kappa evaluation for other perturbations of XLNet in SA 34
Figure 64. Cohen's kappa evaluation for other perturbations of DistilBERT in SS 35
Figure 65. Cohen's kappa evaluation for other perturbations of ELECTRA in SS 35
Figure 66. Cohen's kappa evaluation for other perturbations of Funnel Transformer in SS 36
Figure 67. Cohen's kappa evaluation for other perturbations of XLNet in SS 36

LIST OF TABLES

Table 1. Summary of the characteristics of the four models employed: DistilBERT, ELECTRA, XLNet and Funnel Transformer 37
Table 2. Examples for the SST2 dataset with their corresponding label 37
Table 3. Examples for the CoLA dataset with their corresponding label 37
Table 4. Examples of the PAWS dataset with their corresponding label 38
Table 5. Examples of the MNLI Corpus dataset with their corresponding label 38
Table 6. Examples of the HSOL dataset with their corresponding label 38
Table 7. Summary of the five datasets used: CoLA, HSO, MNLI, SST2 and PAWS 39
Table 8. Summary of the perturbations used with a short example 39
Table 9. Hyperparameters employed in the finetuning phase 40
Table 10. Example of the finetuning scores for DistilBERT in GC 40
Table 11. Best combination of hyperparemeters 40
Table 12. Cohen's kappa scores for character level perturbations in GC 41
Table 13. Cohen's kappa scores for character level perturbations in HSOL 41
Table 14. Cohen's kappa scores for character level perturbations in NLI 42
Table 15. Cohen's kappa scores for character level perturbations in SA 42
Table 16. Cohen's kappa scores for character level perturbations in SS 42
Table 17. Cohen's kappa scores for word level perturbations in GC 43
Table 18. Cohen's kappa scores for word level perturbations in HSOL 43
Table 19. Cohen's kappa scores for word level perturbations in NLI 43
Table 20. Cohen's kappa scores for word level perturbations in SA 44
Table 21. Cohen's kappa scores for word level perturbations in SS 44
Table 22. Cohen's kappa scores for other perturbations in GC 45
Table 23. Cohen's kappa scores for other perturbations in HSOL 45
Table 24. Cohen's kappa scores for other perturbations in NLI 46
Table 25. Cohen's kappa scores for other perturbations in SA 46
Table 26. Cohen's kappa scores for other perturbations in SS 46
Table 27. Average kappa score per model 47
Table 28. Average kappa score per NLP task 47
Table 29. Average kappa score per perturbation type 47
Table 30. Average kappa score per character perturbation 47
Table 31. Average kappa score per word perturbation 47
Table 32. Average kappa score per other type of perturbation 48
Table 33. Average kappa score per perturbation type when non-represented perturbations are removed 48

Figures

Example of distribution shift

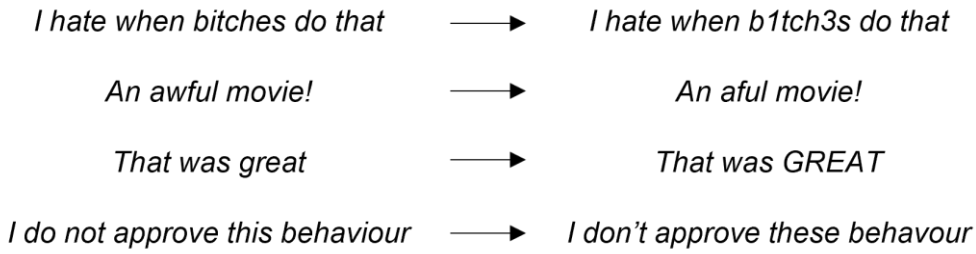

Figure 1. Example of distribution shifts

Attention matrix

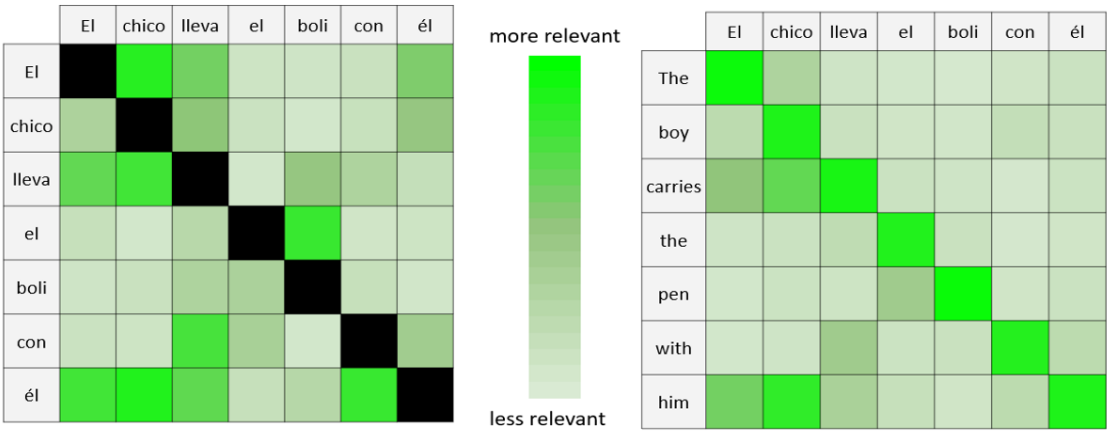

Figure 2. On the left, visual representation of self-attention. On the right, visual representation of attention. Attention matrixes help visualise syntactic relationships, such as the dependency of the subject ("el chico") and their passive mention ("con él")

Models

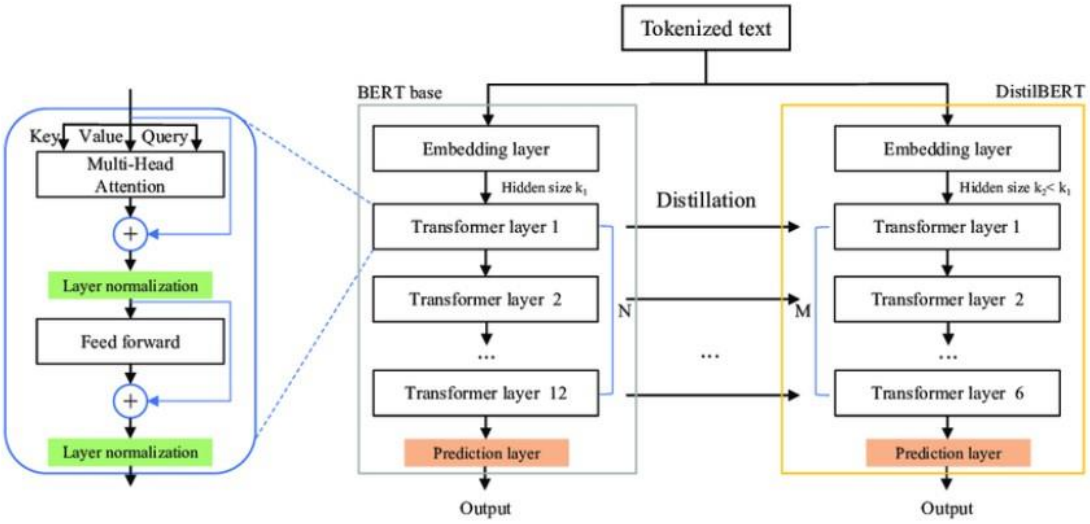

Figure 3. The DistilBERT model and its components. Source: (Adel, et al., 2022)

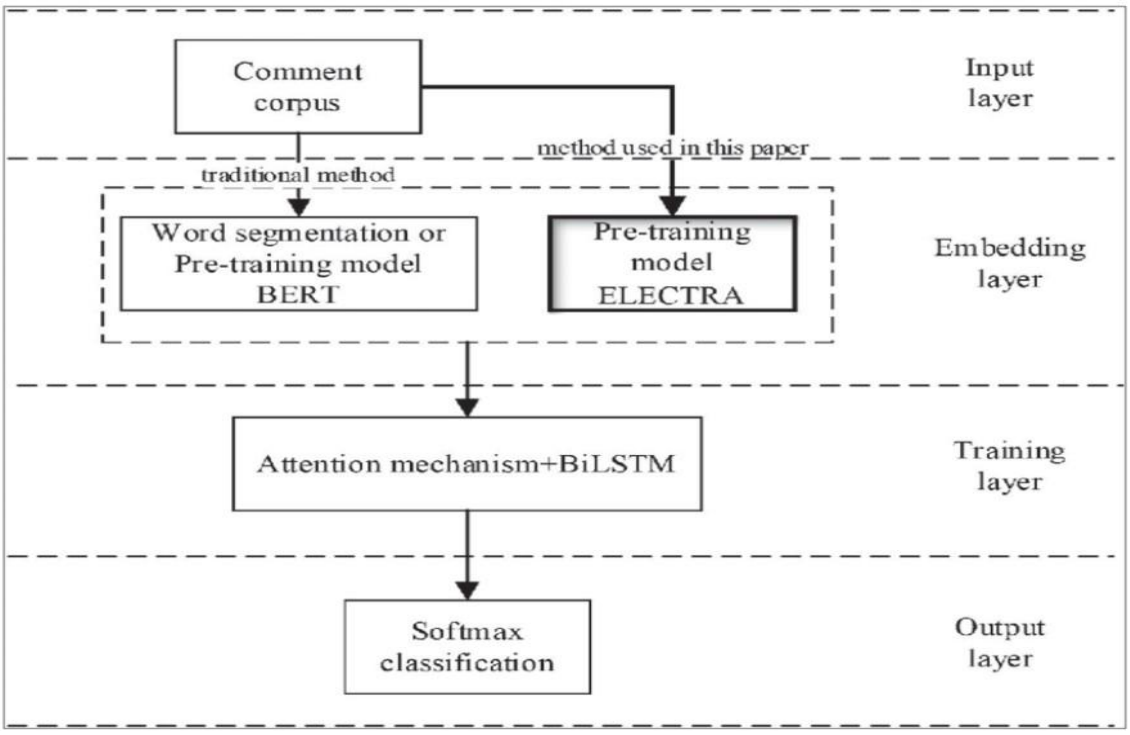

Figure 4. The ELECTRA model and its components. Source: (Zhang, Yu, & Zhu, 2021)

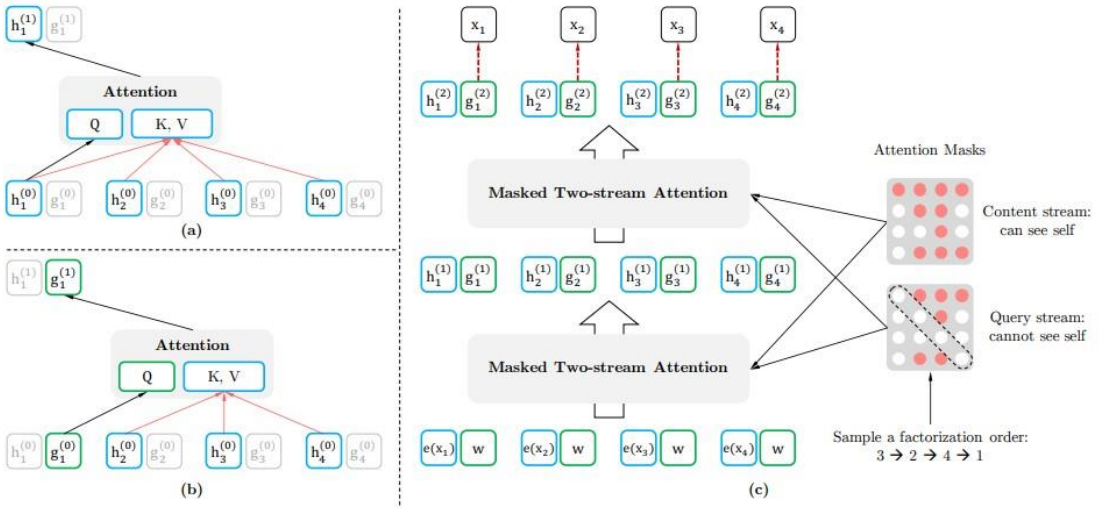

Figure 5. The XLNet model and its components. Source: (Yang, et al., 2019)

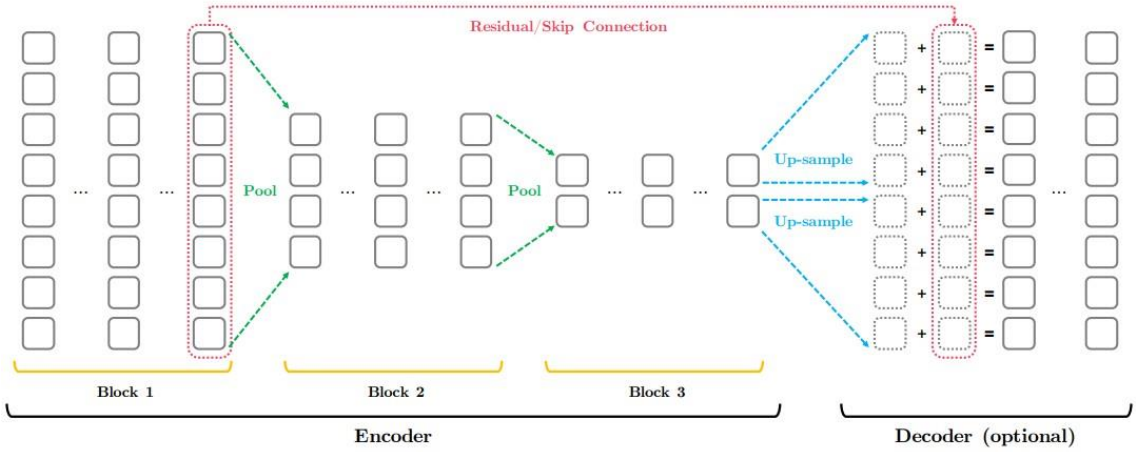

Figure 6. The funnel transformer model and its components. Source: (Zihang, Guokun, Yiming, & Quoc, 2020)

		Rater 1		
		Class A	Class B	Total
Rater 2	Class A	40	10	50
	Class B	20	30	50
	Total	60	40	100

Figure 7. Confusion matrix to explain the computation of the Kappa Cohen's score

Evaluation

Character level perturbations

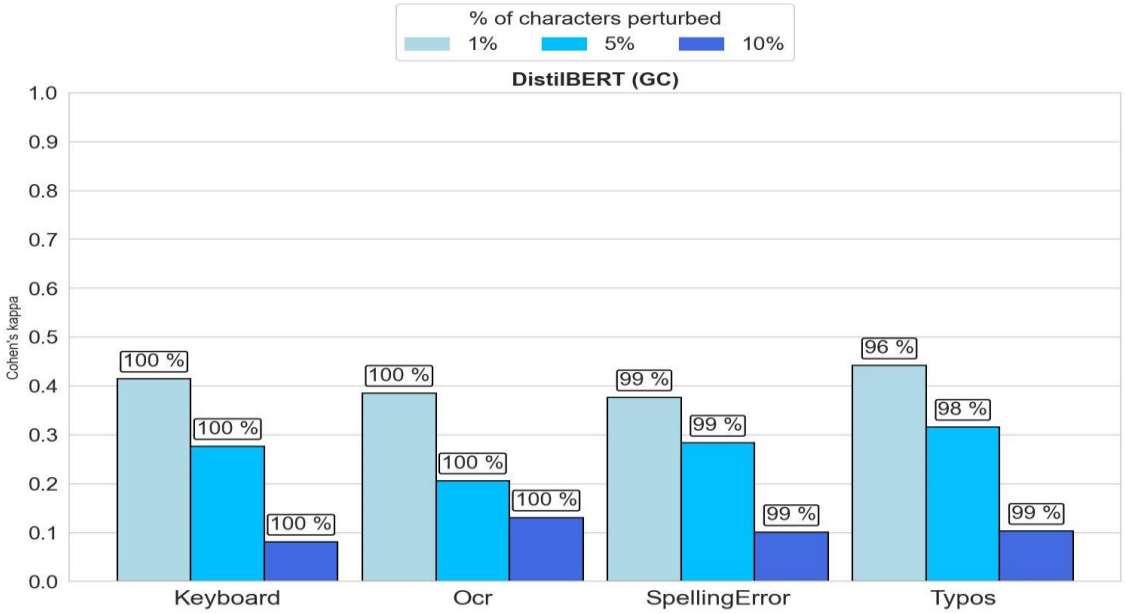

Figure 8. Cohen's kappa values for character level perturbations per perturbation percentage using DistilBERT in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

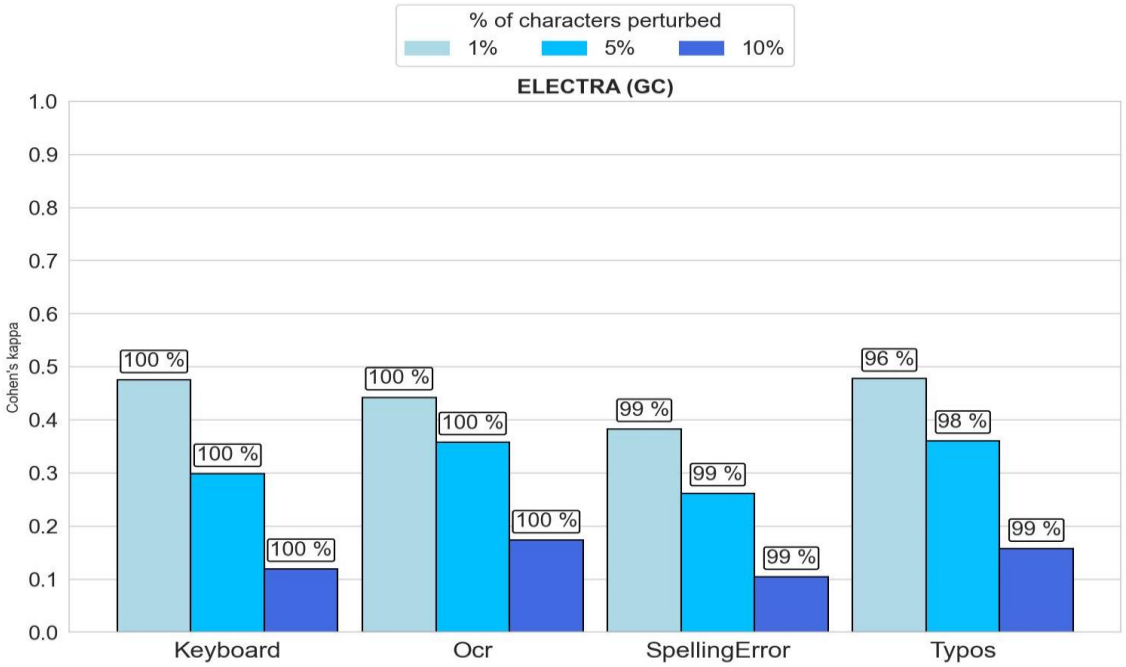

Figure 9. Cohen's kappa values for character level perturbations per perturbation percentage using ELECTRA in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

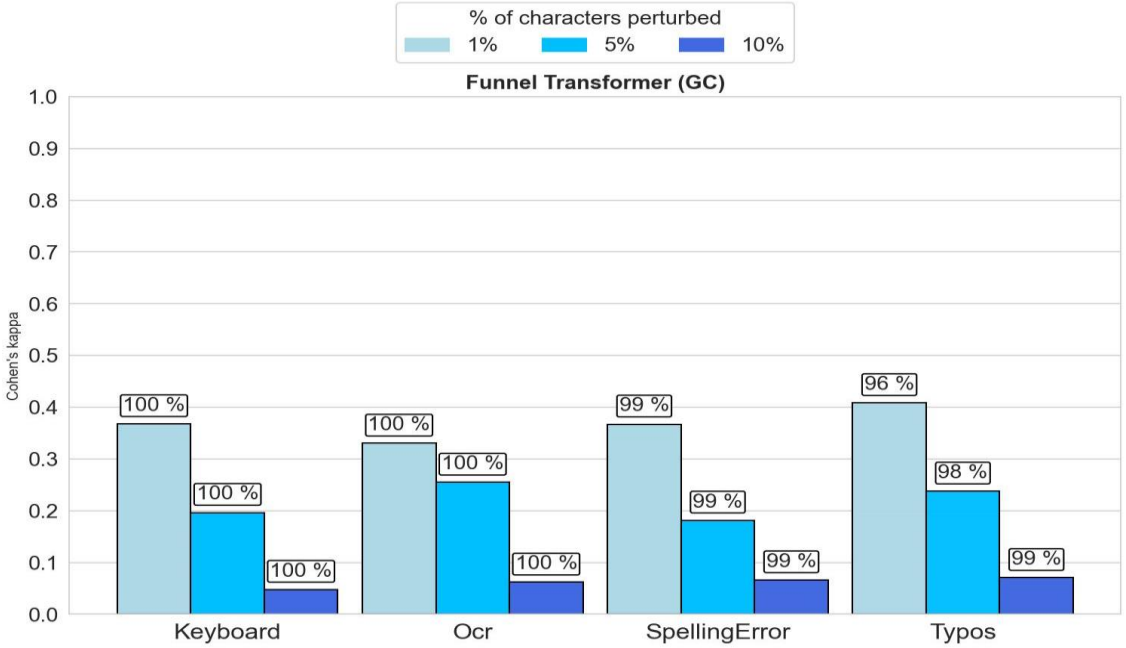

Figure 10. Cohen's kappa values for character level perturbations per perturbation percentage using Funnel Transformer in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

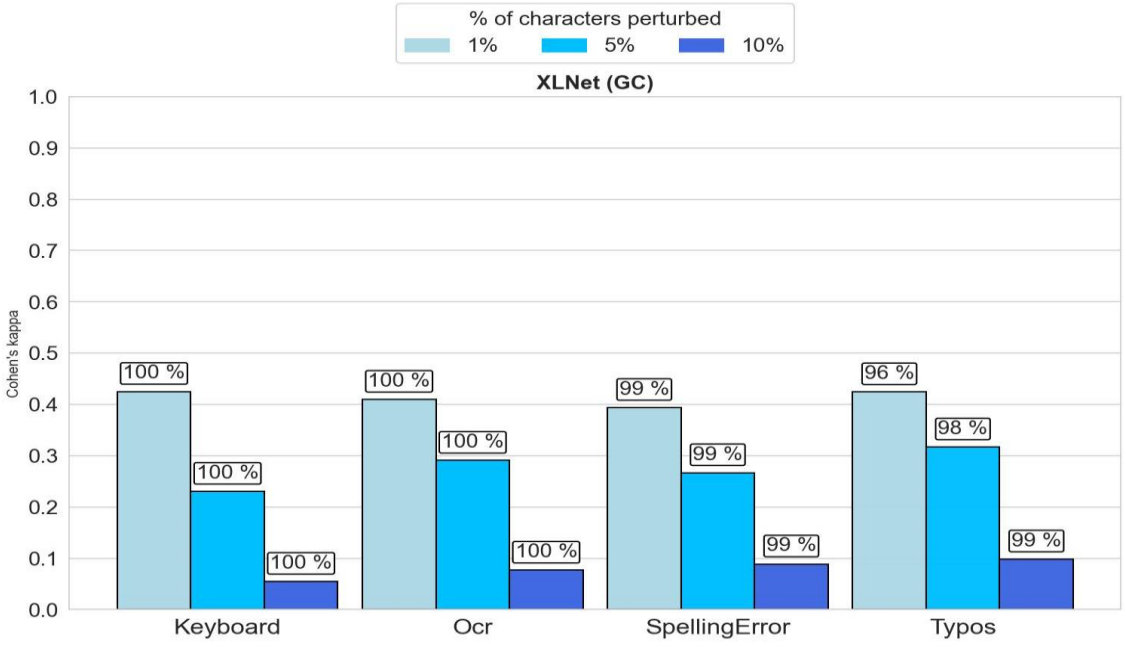

Figure 11. Cohen's kappa values for character level perturbations per perturbation percentage using XLNet in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

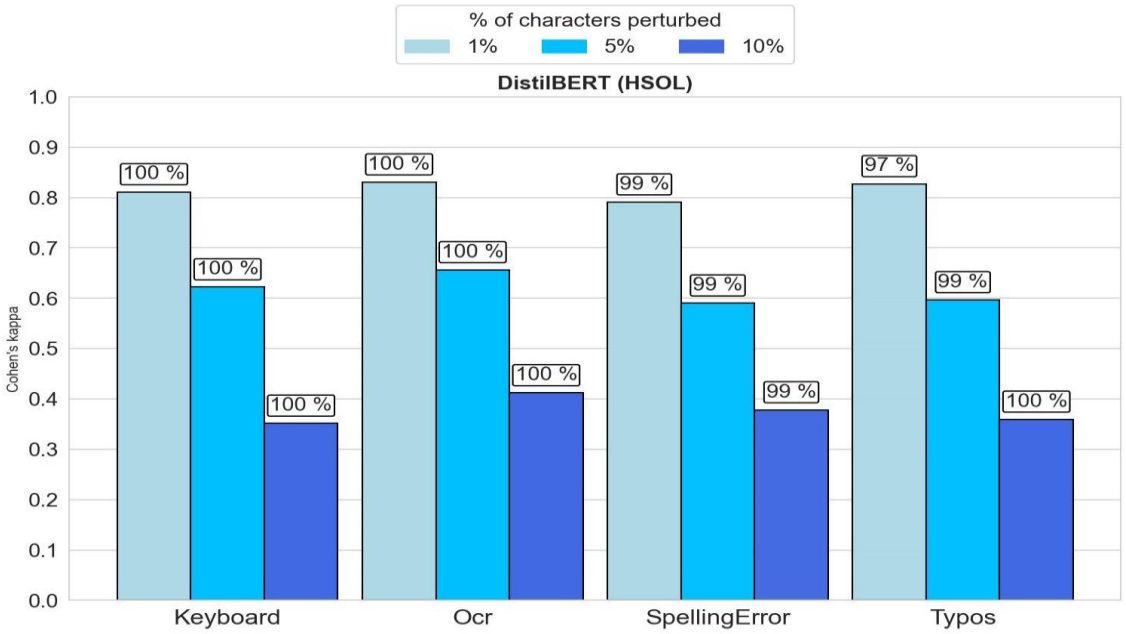

Figure 12. Cohen's kappa values for character level perturbations per perturbation percentage using DistilBERT in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

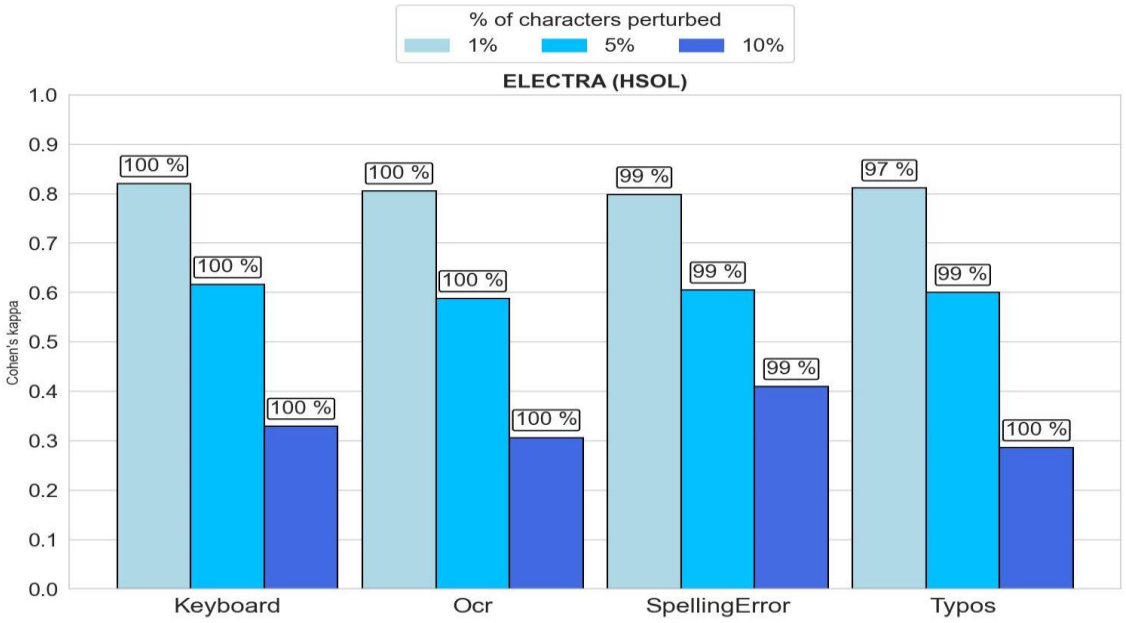

Figure 13. Cohen's kappa values for character level perturbations per perturbation percentage using ELECTRA in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

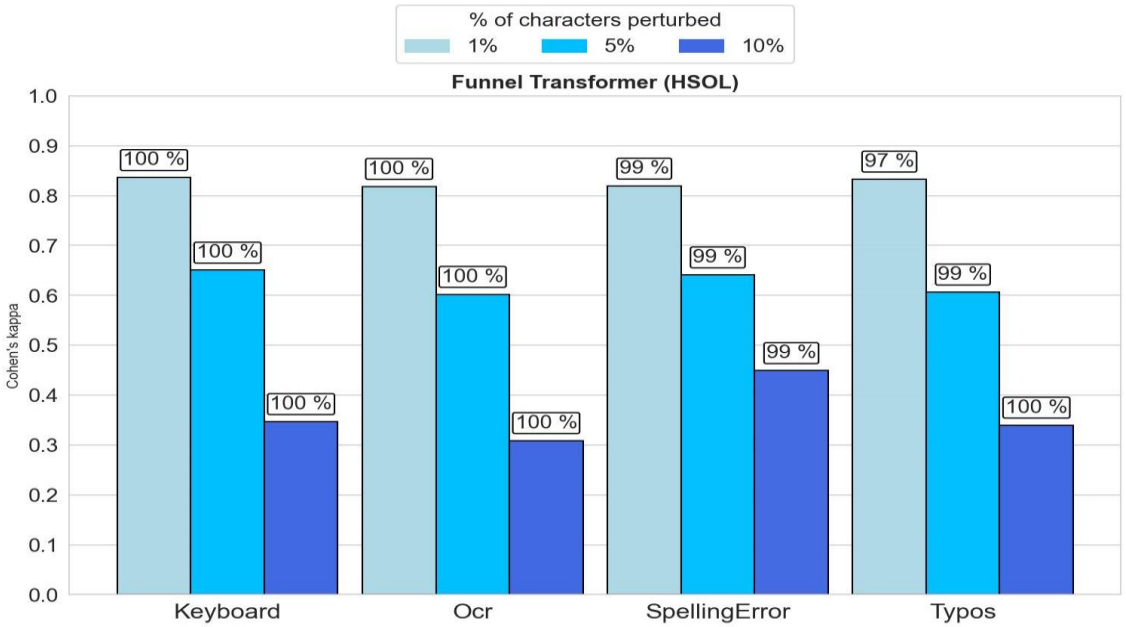

Figure 14. Cohen's kappa values for character level perturbations per perturbation percentage using Funnel Transformer in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

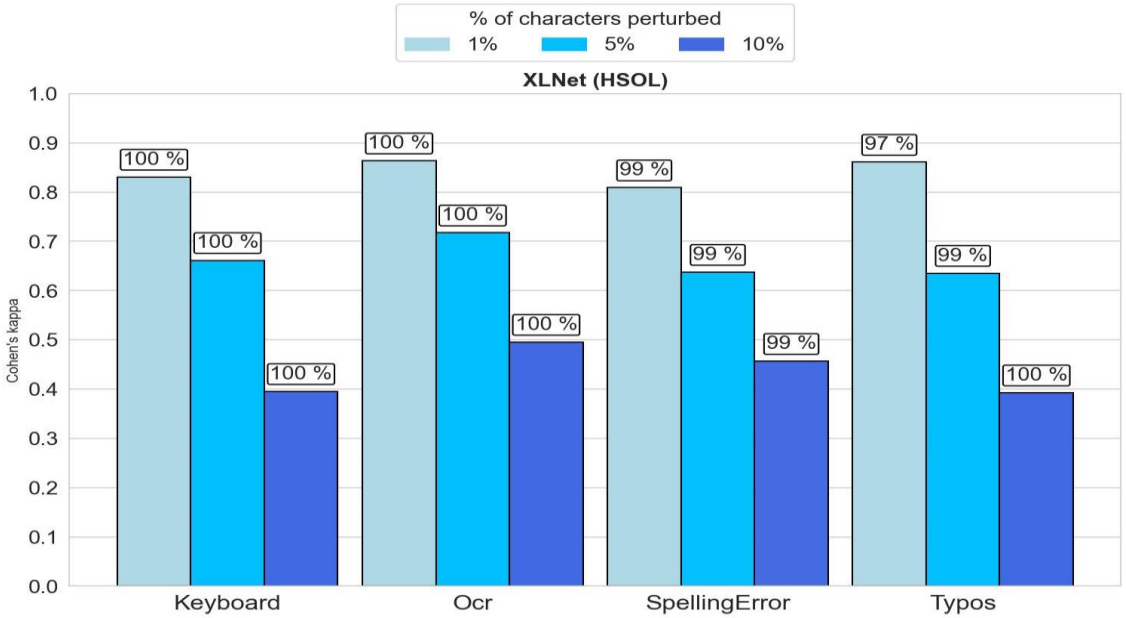

Figure 15. Cohen's kappa values for character level perturbations per perturbation percentage using XLNet in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

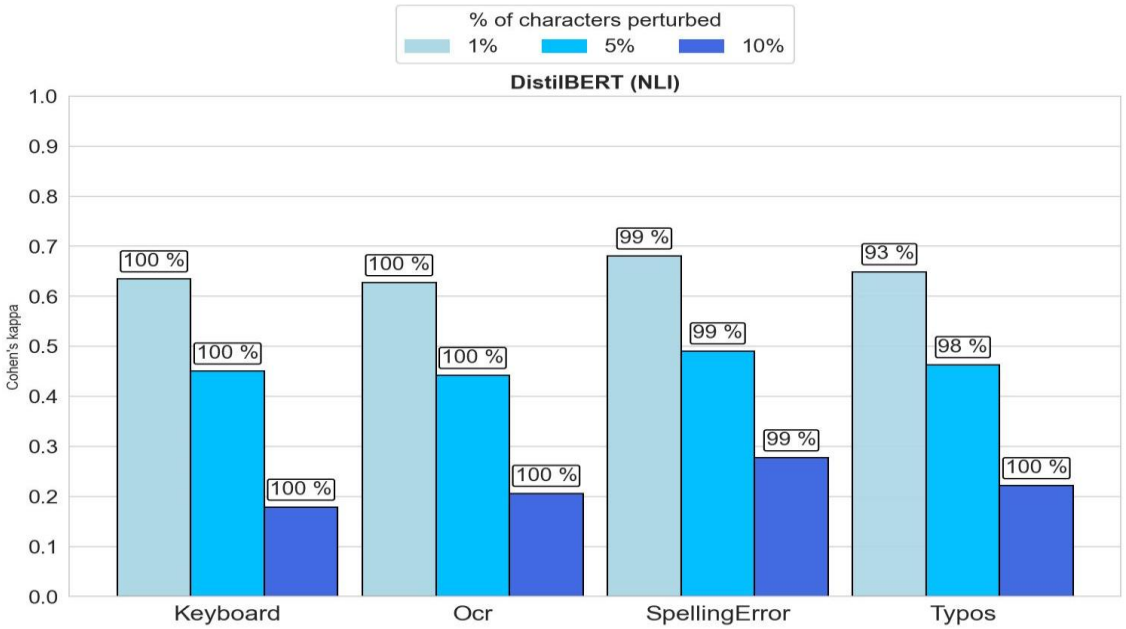

Figure 16. Cohen's kappa values for character level perturbations per perturbation percentage using DistilBERT in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

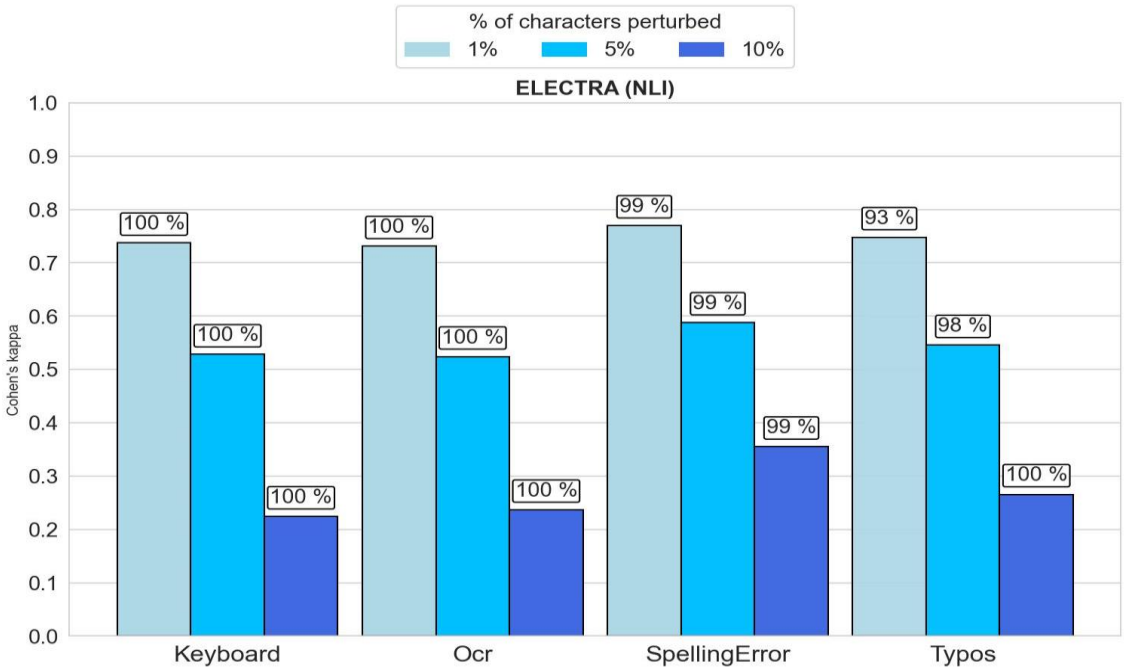

Figure 17. Cohen's kappa values for character level perturbations per perturbation percentage using ELECTRA in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

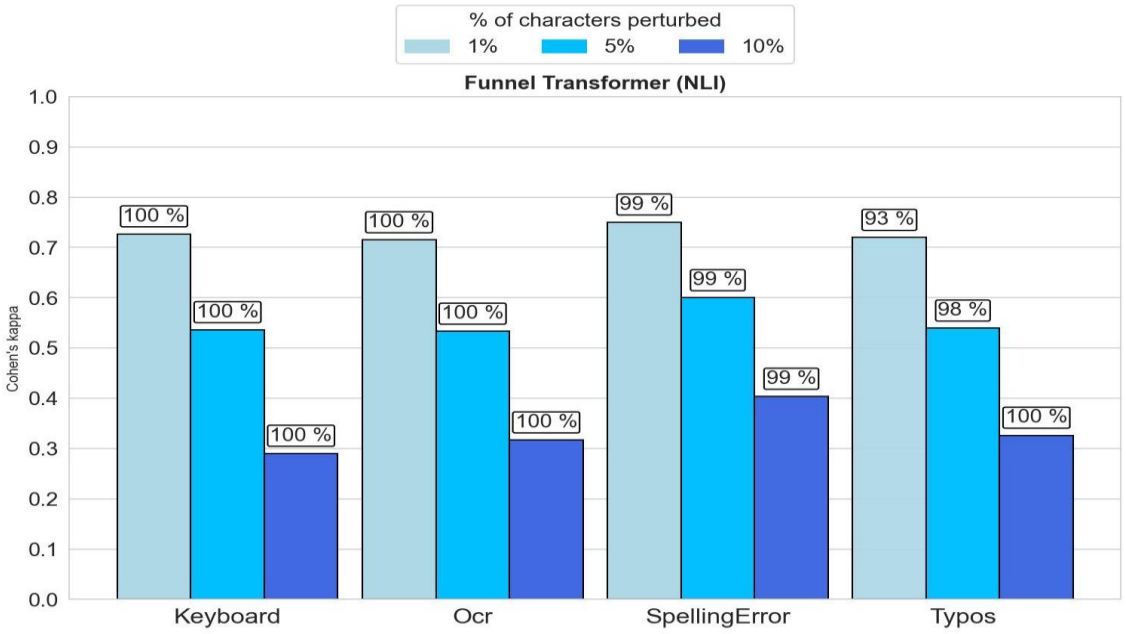

Figure 18. Cohen's kappa values for character level perturbations per perturbation percentage using Funnel Transformer in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

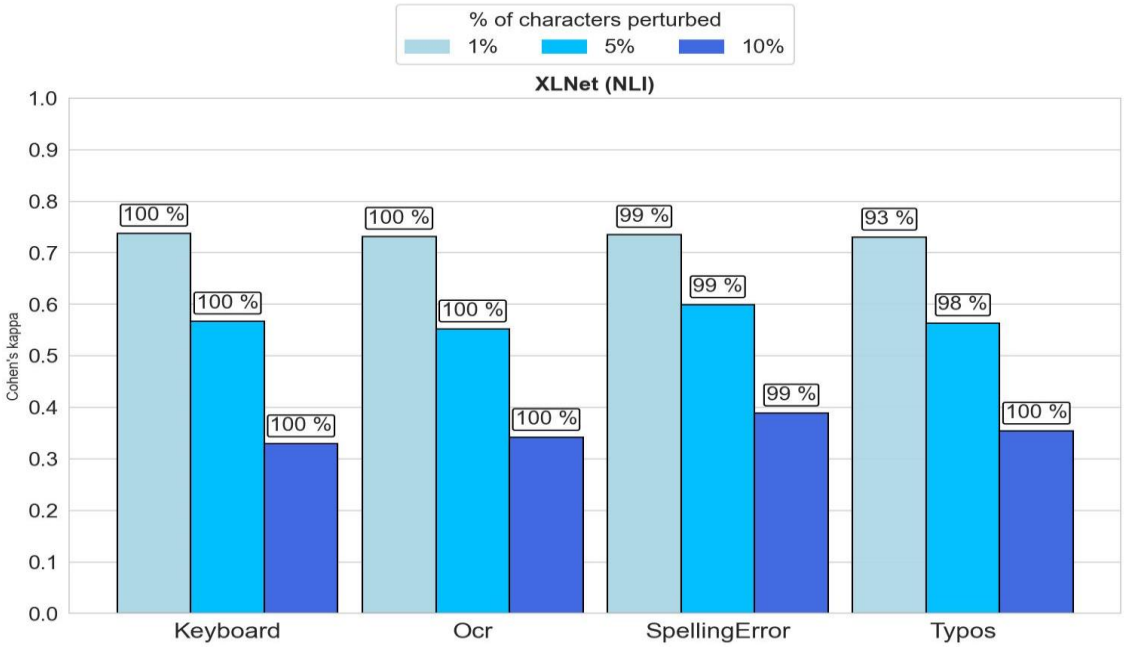

Figure 19. Cohen's kappa values for character level perturbations per perturbation percentage using XLNet in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

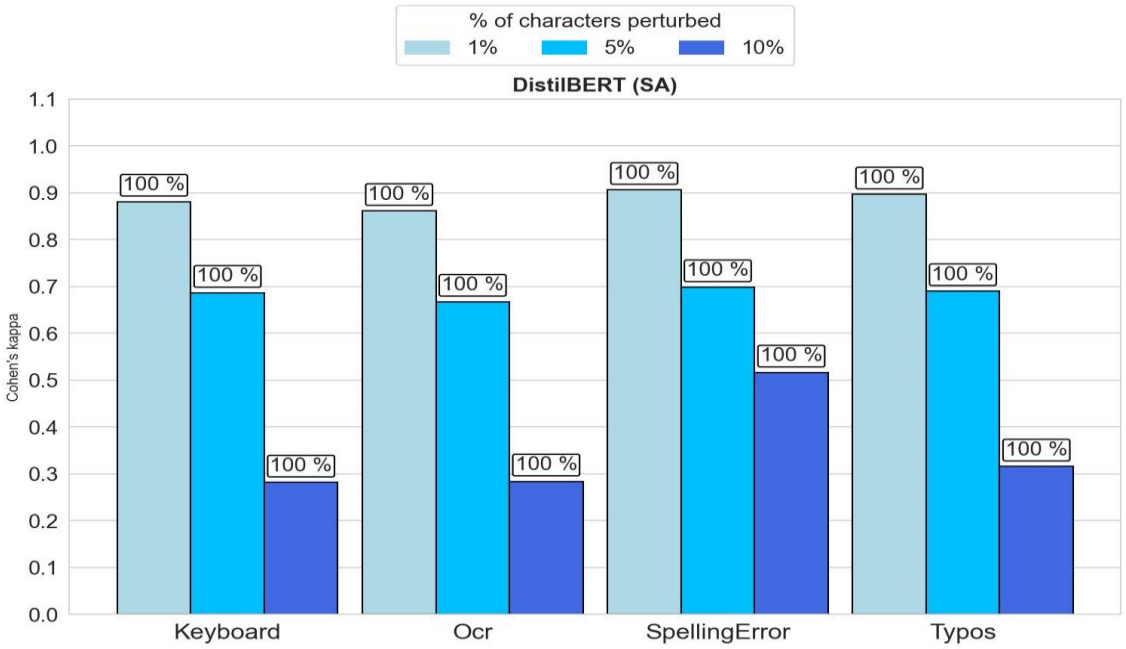

Figure 20. Cohen's kappa values for character level perturbations per perturbation percentage using DistilBERT in the Sentiment Analysis task. The values above the bars show the proportion of perturbed samples for that given perturbation

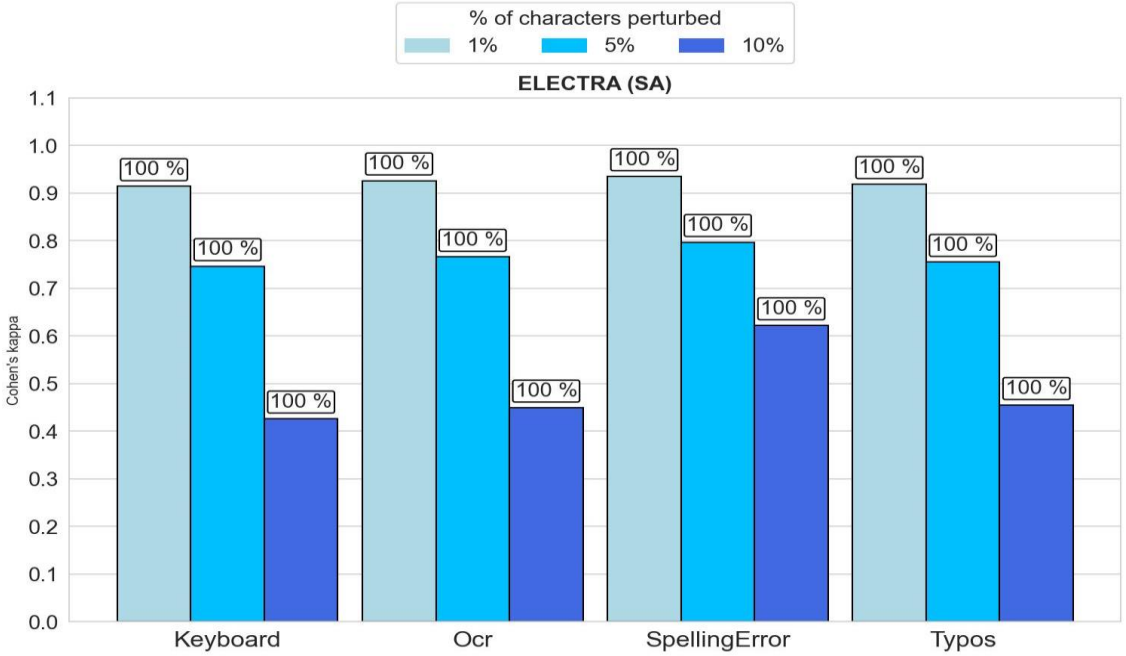

Figure 21. Cohen's kappa values for character level perturbations per perturbation percentage using ELECTRA in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

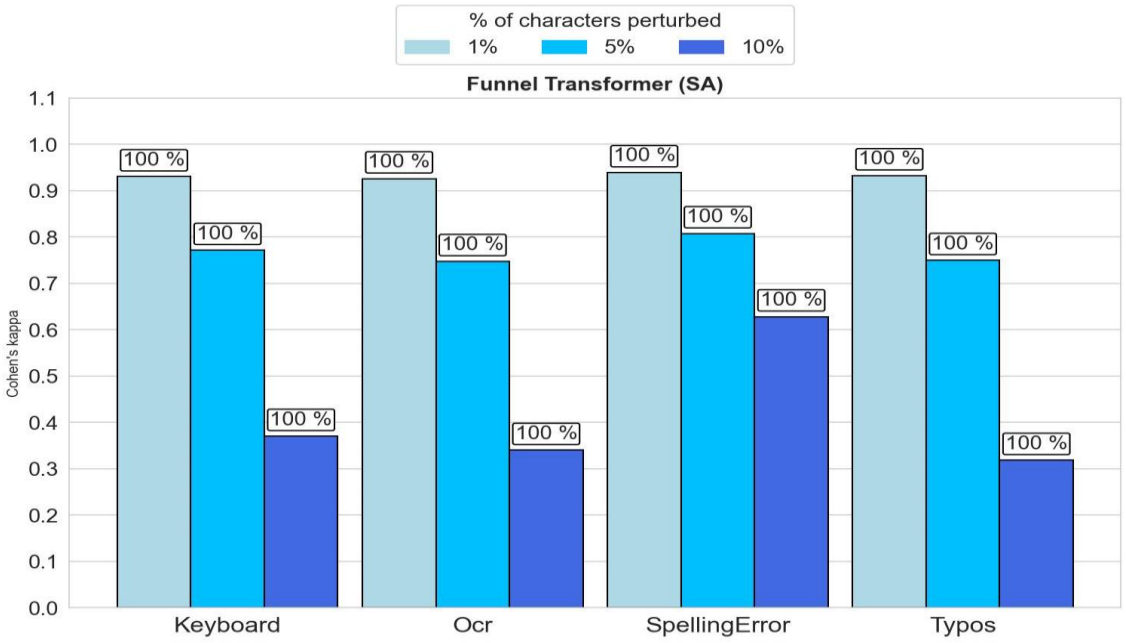

Figure 22. Cohen's kappa values for character level perturbations per perturbation percentage using Funnel Transformer in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

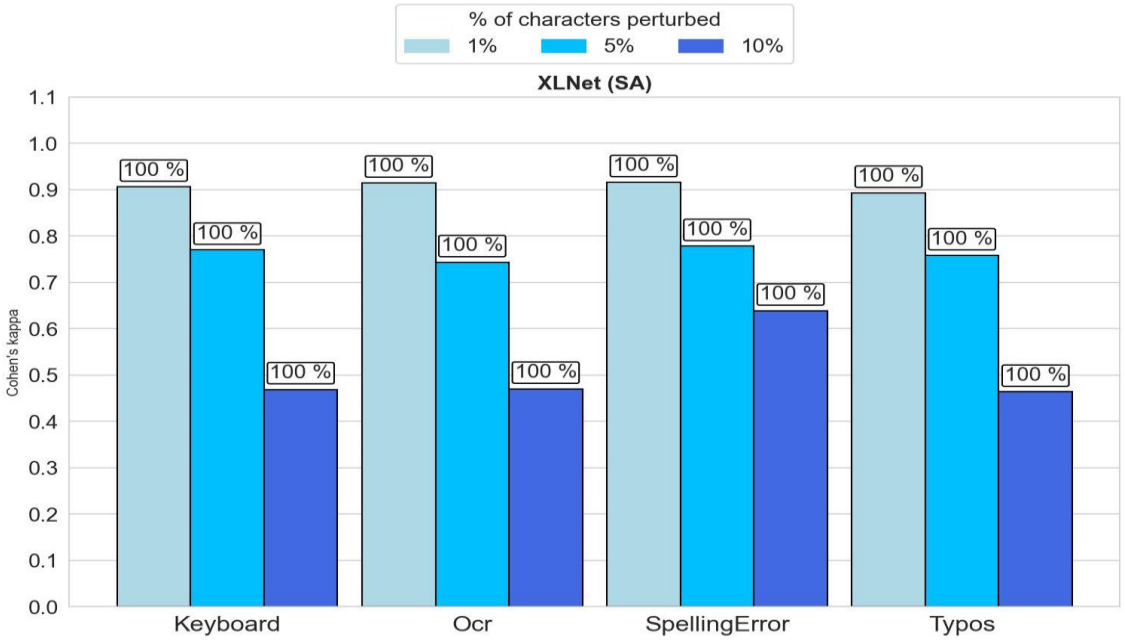

Figure 23. Cohen's kappa values for character level perturbations per perturbation percentage using XLNet in the Sentiment Analysis task. The values above the bars show the proportion of perturbed samples for that given perturbation

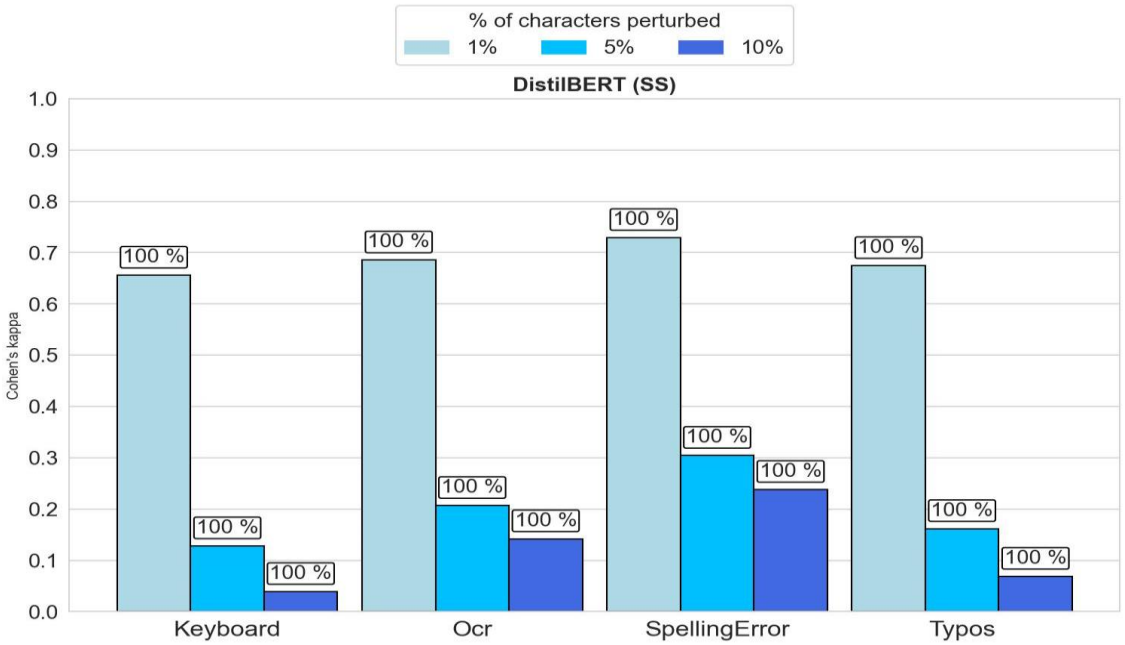

Figure 24. Cohen's kappa values for character level perturbations per perturbation percentage using DistilBERT in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

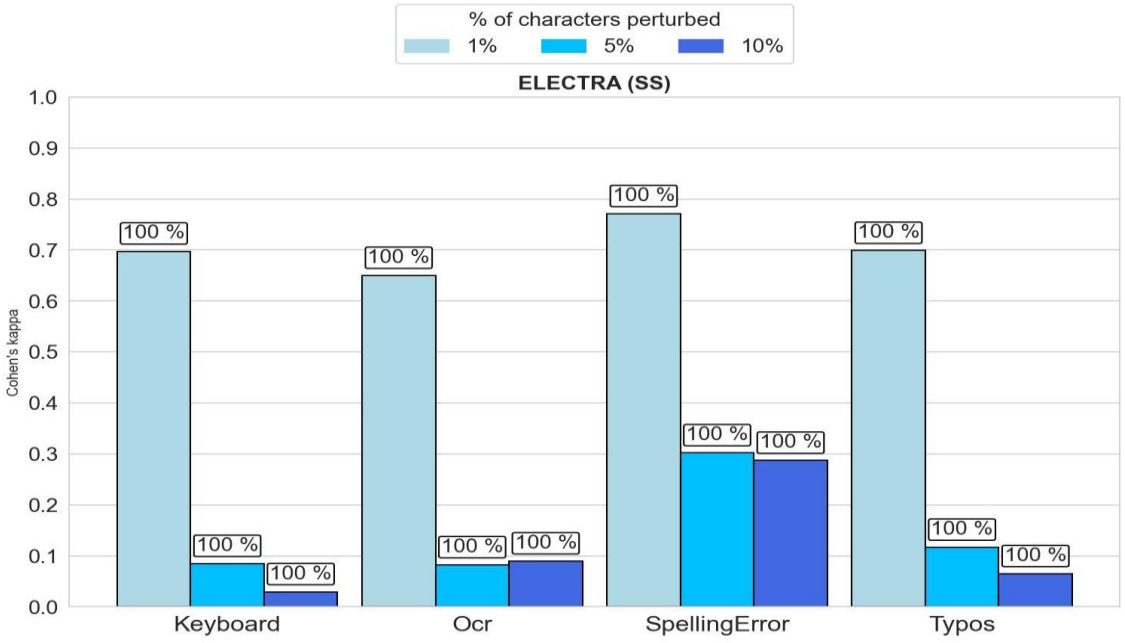

Figure 25. Cohen's kappa values for character level perturbations per perturbation percentage using ELECTRA in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

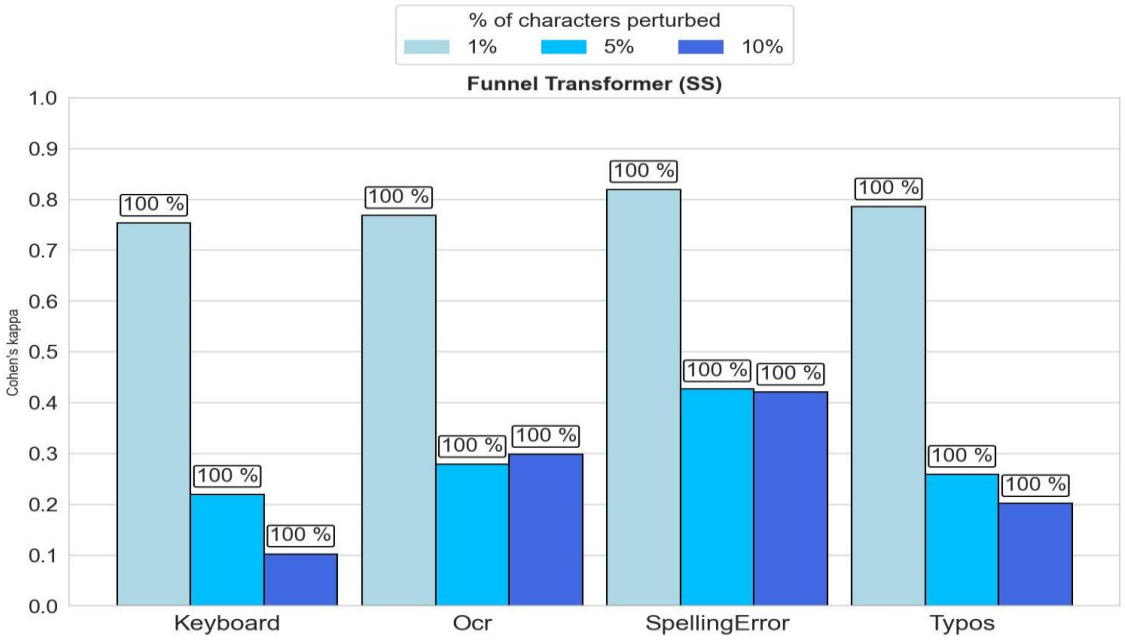

Figure 26. Cohen's kappa values for character level perturbations per perturbation percentage using Funnel Transformer in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

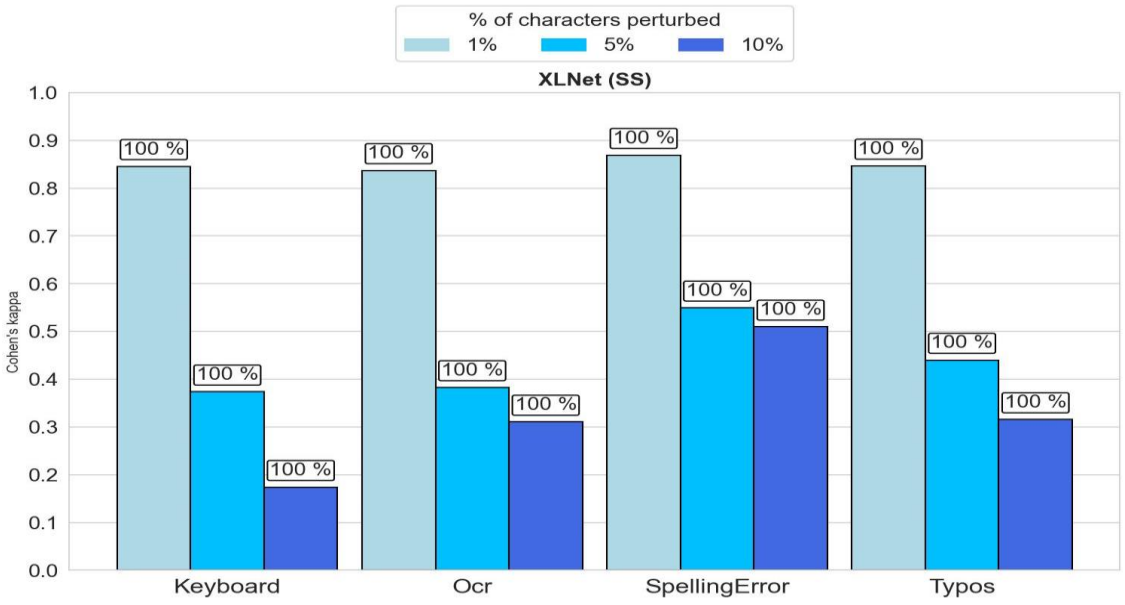

Figure 27. Cohen's kappa values for character level perturbations per perturbation percentage using XLNet in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

Word level perturbations

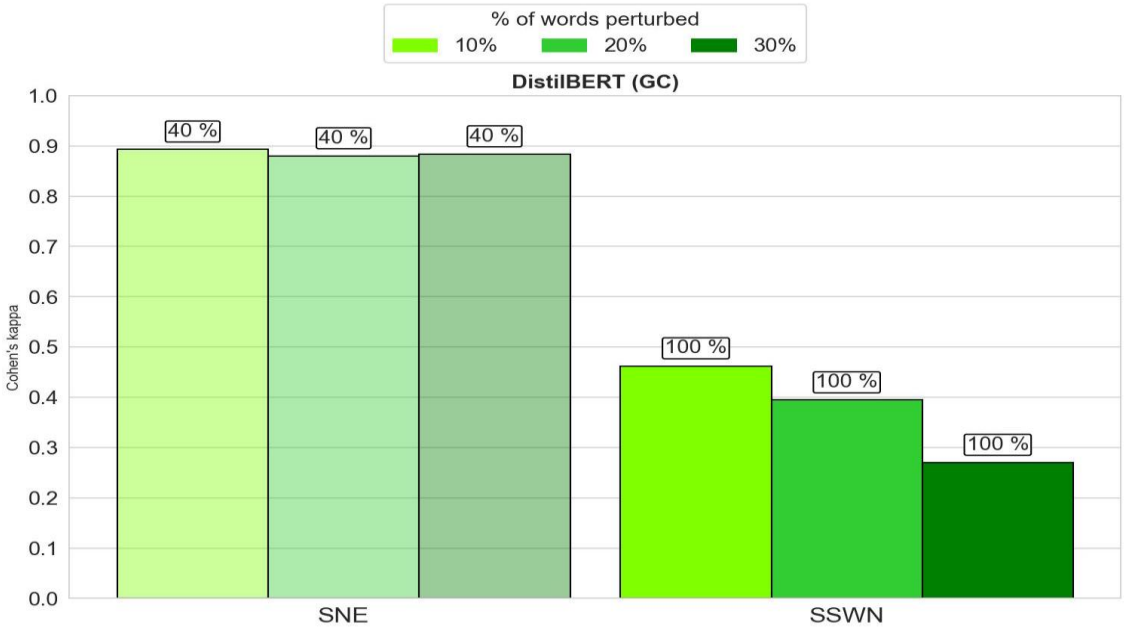

Figure 28. Cohen's kappa values for word level perturbations per perturbation percentage using DistilBERT in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

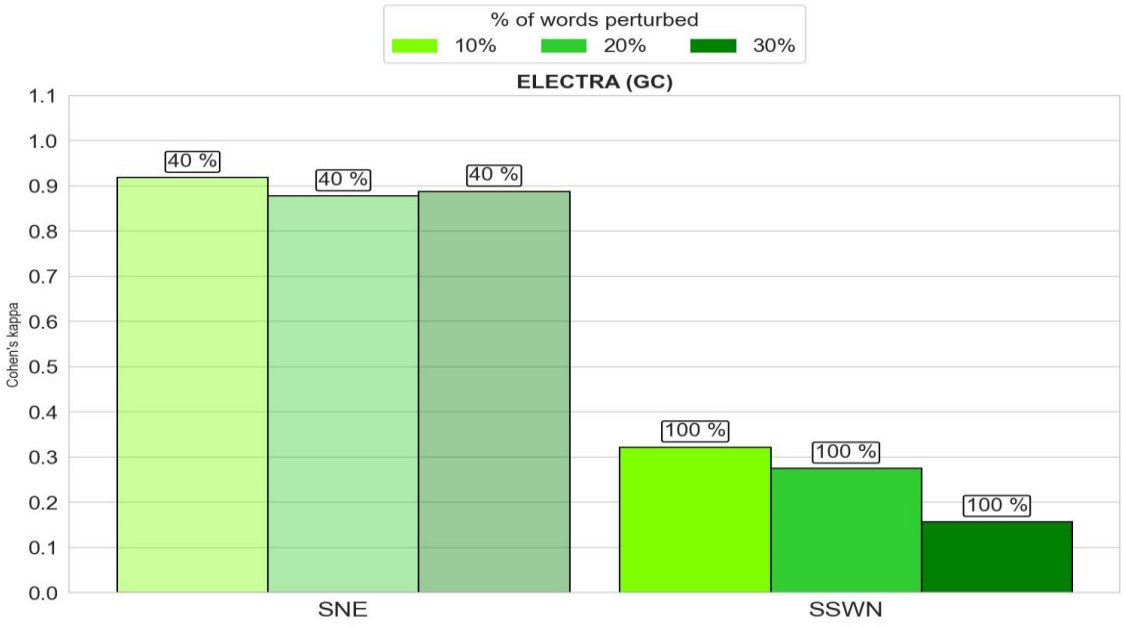

Figure 29. Cohen's kappa values for word level perturbations per perturbation percentage using ELECTRA in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

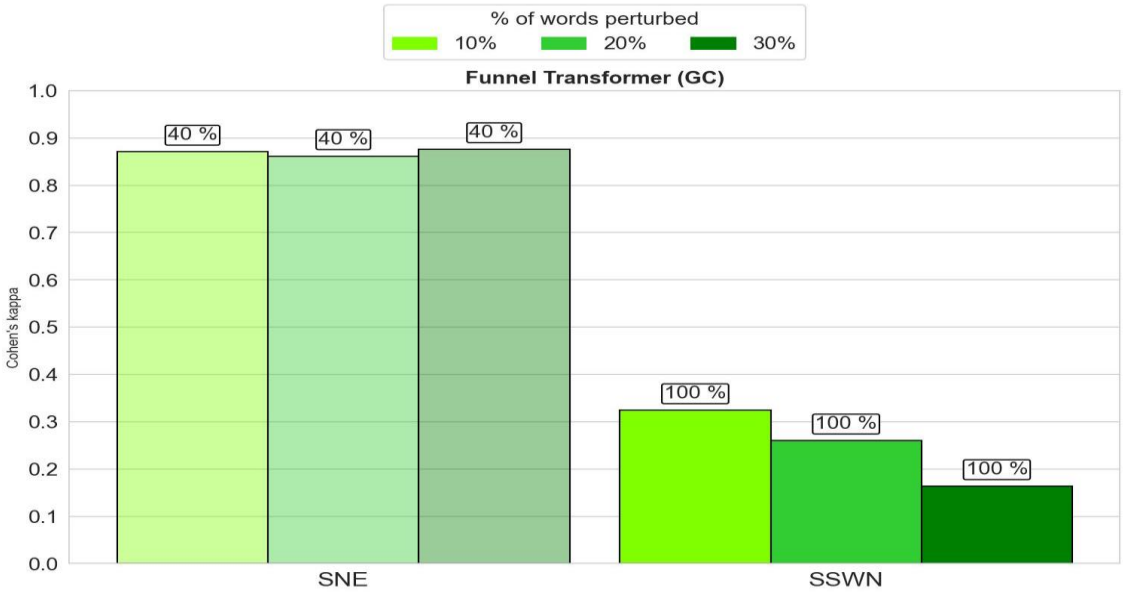

Figure 30. Cohen's kappa values for word level perturbations per perturbation percentage using Funnel Transformer in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

Figure 31. Cohen's kappa values for word level perturbations per perturbation percentage using XLNet in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

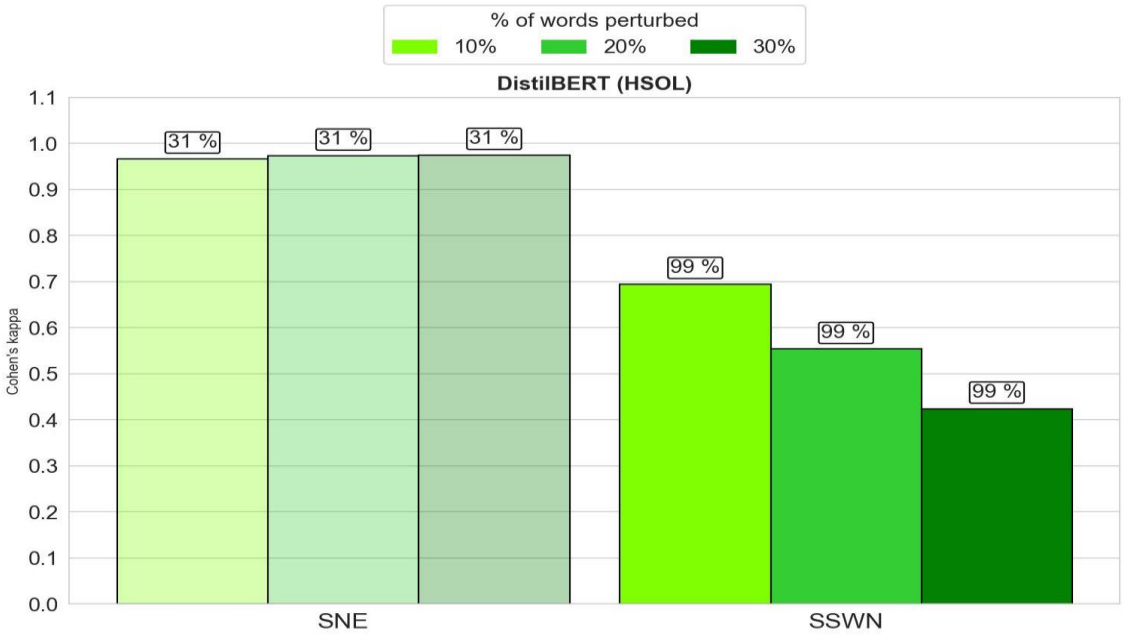

Figure 32. Cohen's kappa values for word level perturbations per perturbation percentage using DistilBERT in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

Figure 33. Cohen's kappa values for word level perturbations per perturbation percentage using ELECTRA in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

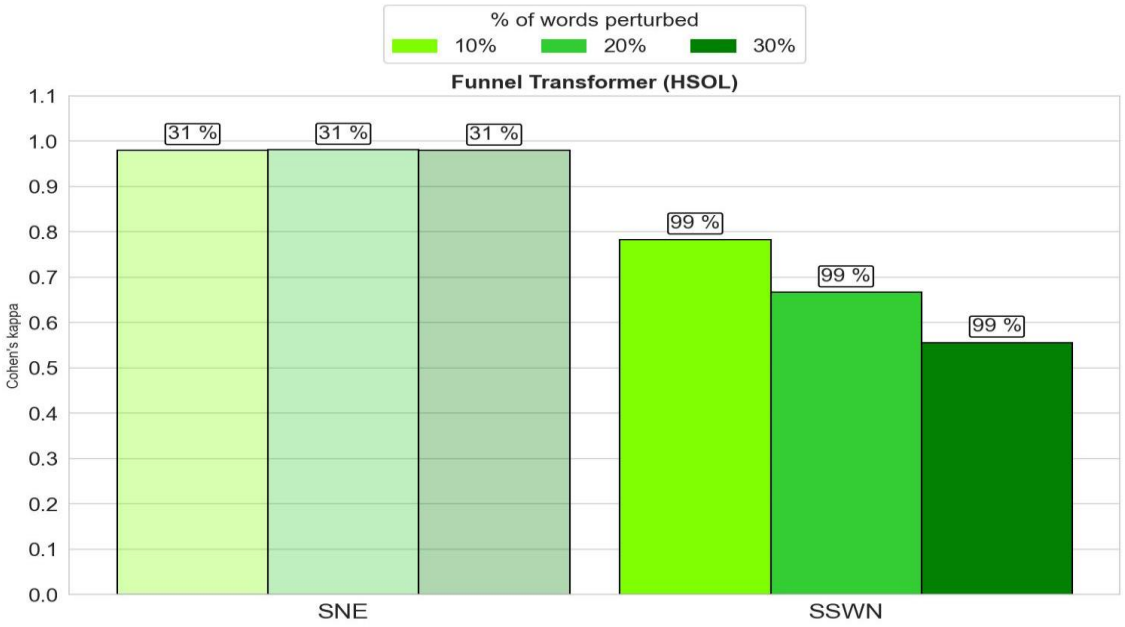

Figure 34. Cohen's kappa values for word level perturbations per perturbation percentage using Funnel Transformer in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

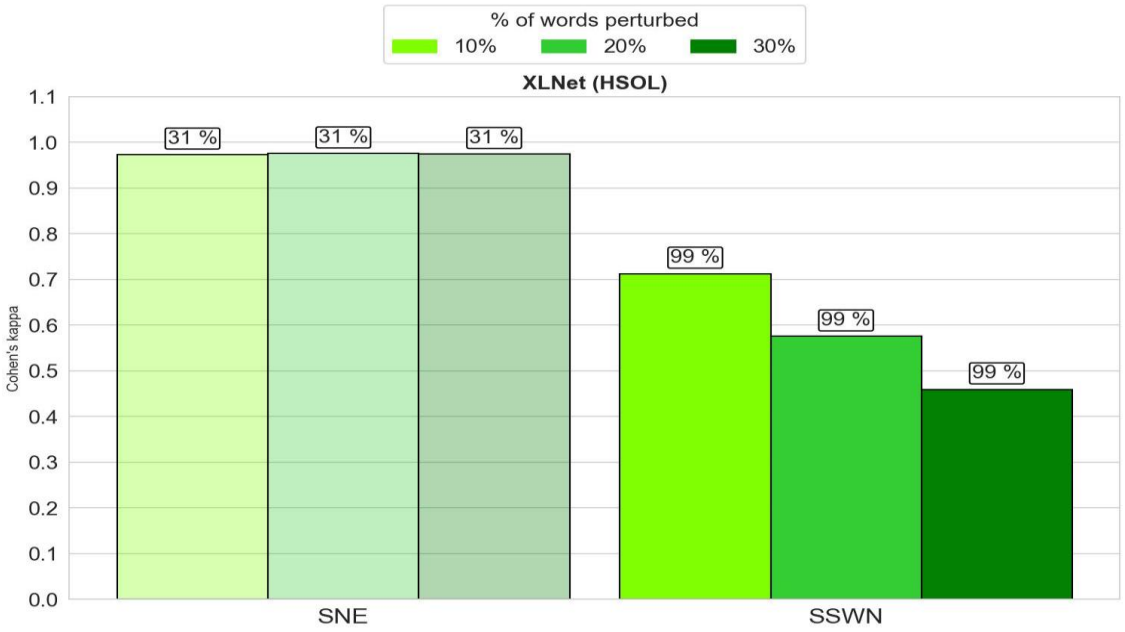

Figure 35. Cohen's kappa values for word level perturbations per perturbation percentage using XLNet in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

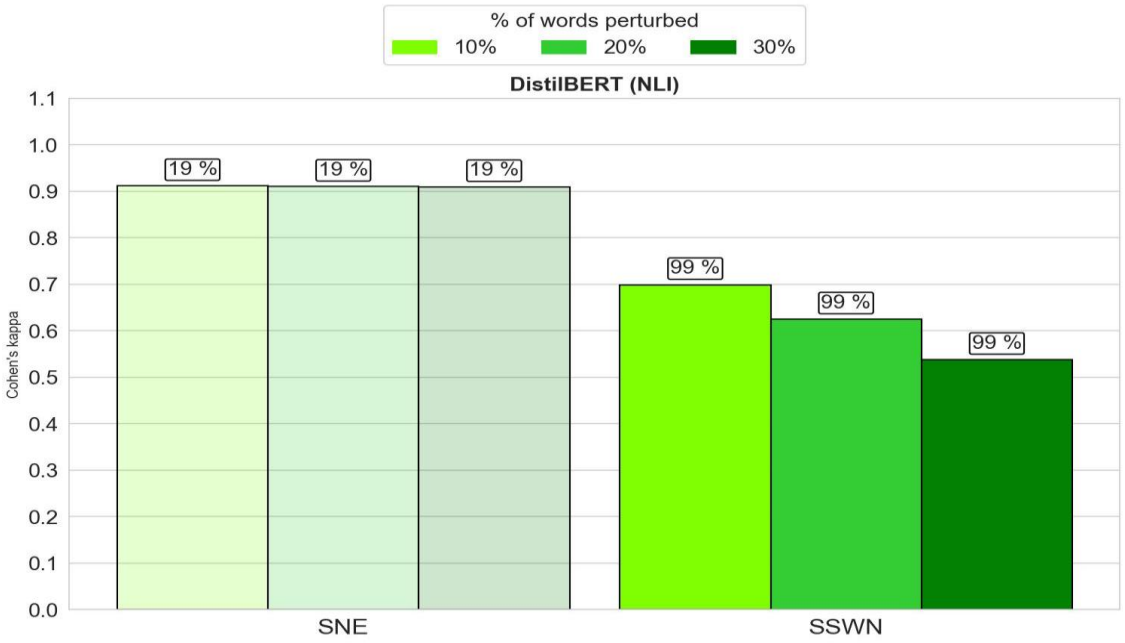

Figure 36. Cohen's kappa values for word level perturbations per perturbation percentage using DistilBERT in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

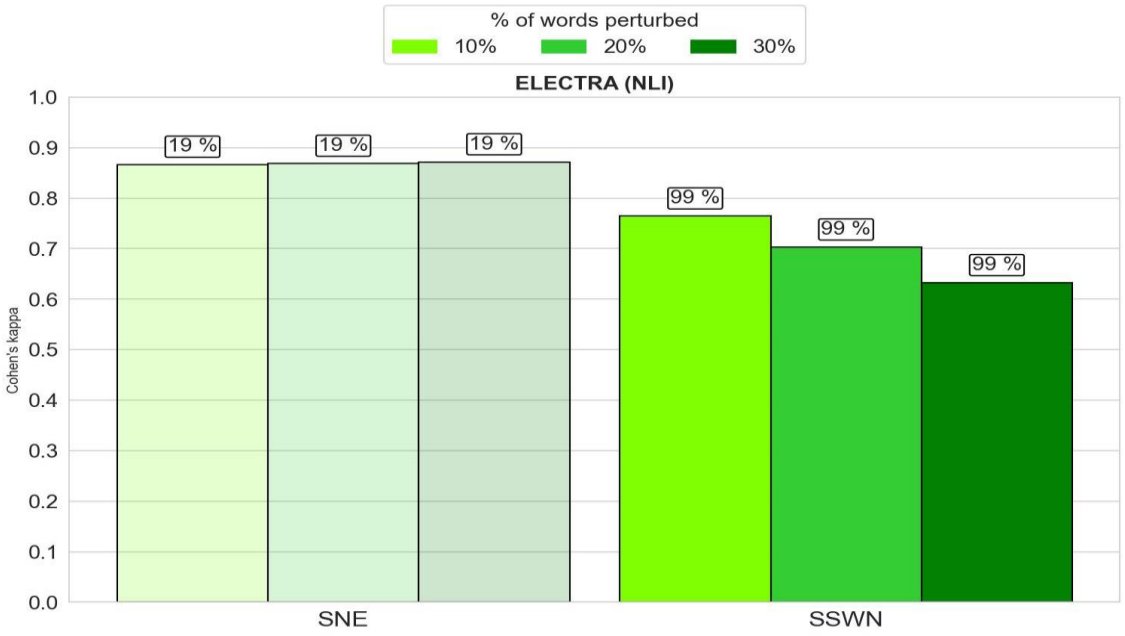

Figure 37. Cohen's kappa values for word level perturbations per perturbation percentage using ELECTRA in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

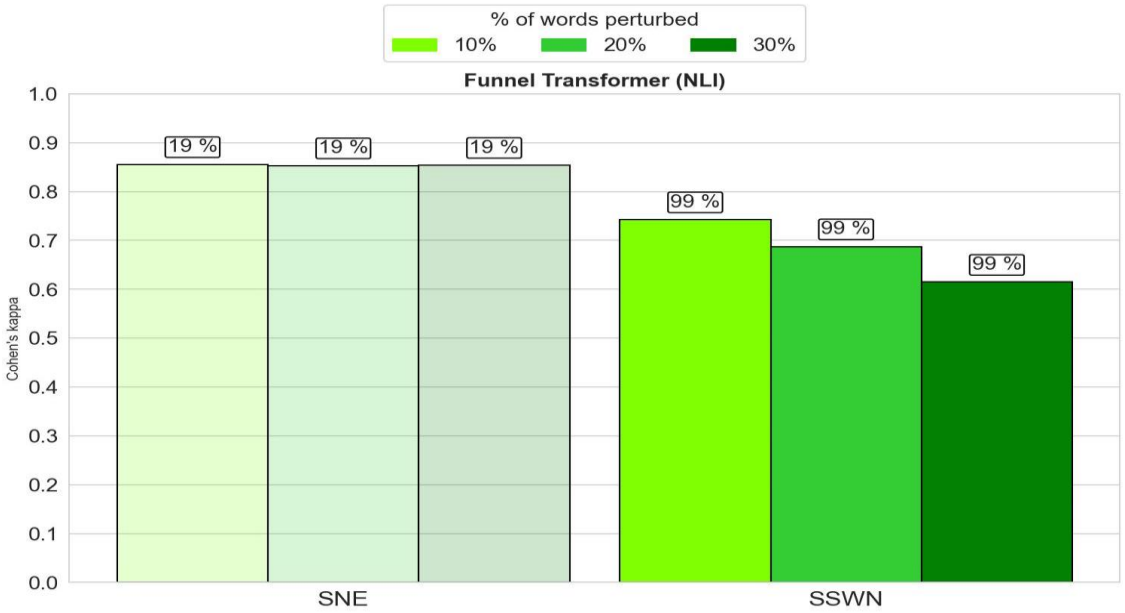

Figure 38. Cohen's kappa values for word level perturbations per perturbation percentage using Funnel Transformer in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

Figure 39. Cohen's kappa values for word level perturbations per perturbation percentage using XLNet in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

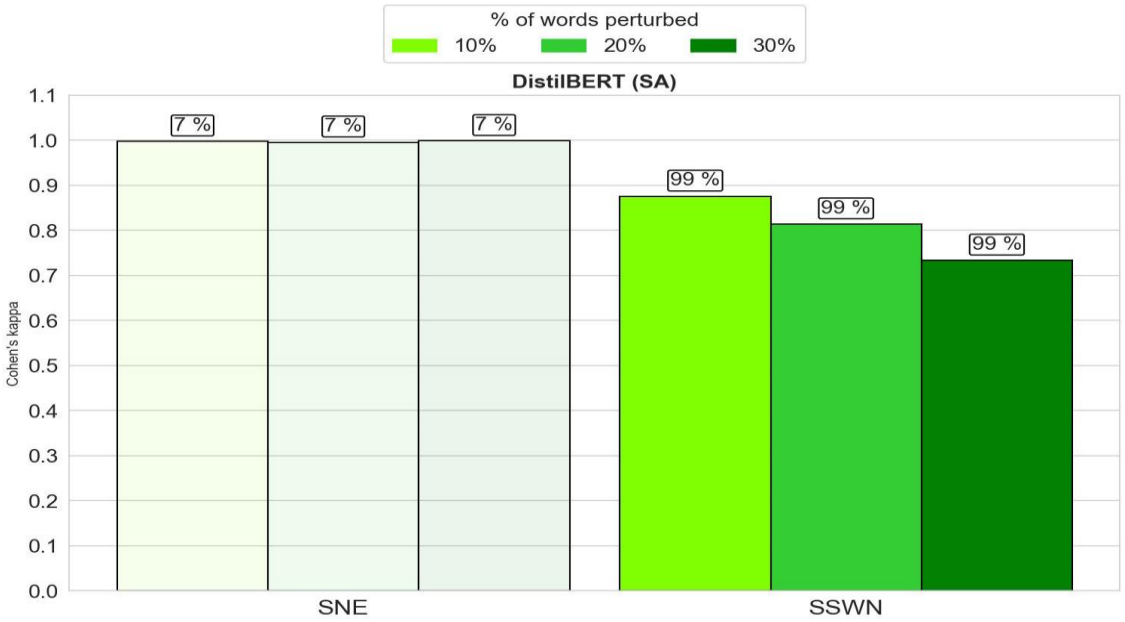

Figure 40. Cohen's kappa values for word level perturbations per perturbation percentage using DistilBERT in the Sentiment Analysis task. The values above the bars show the proportion of perturbed samples for that given perturbation

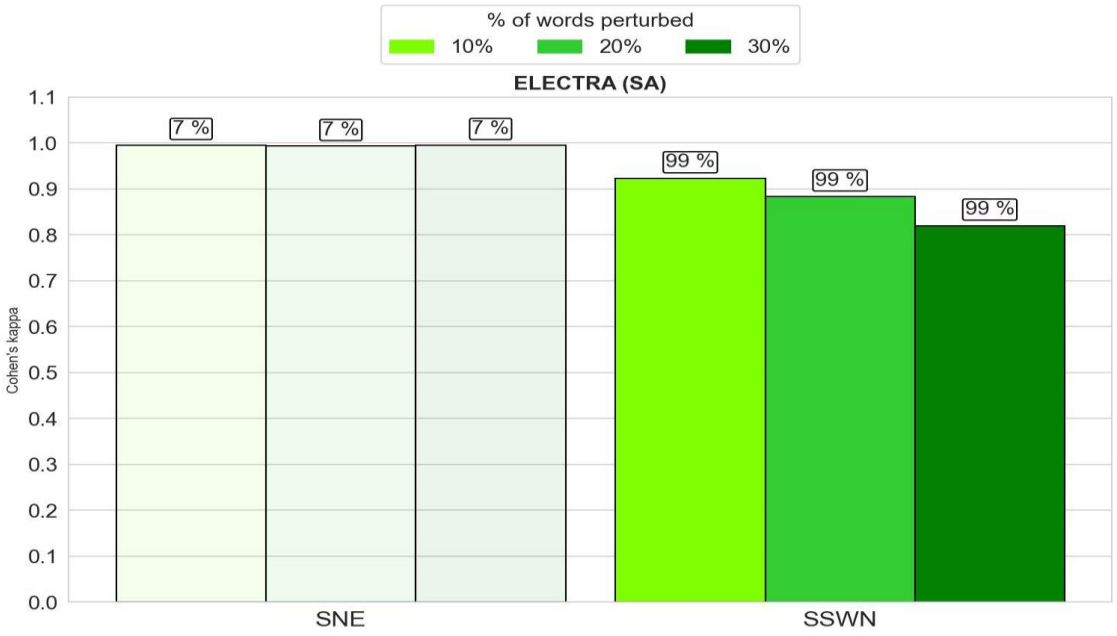

Figure 41. Cohen's kappa values for word level perturbations per perturbation percentage using ELECTRA in the Sentiment Analysis task. The values above the bars show the proportion of perturbed samples for that given perturbation

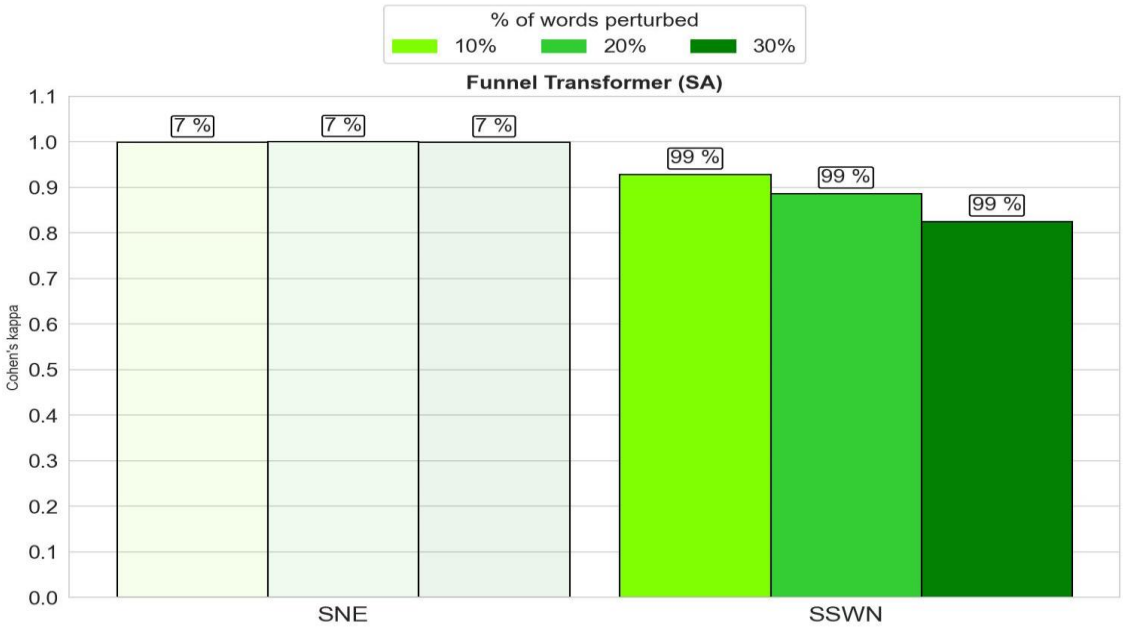

Figure 42. Cohen's kappa values for word level perturbations per perturbation percentage using Funnel Transformer in the Sentiment Analysis task. The values above the bars show the proportion of perturbed samples for that given perturbation

Figure 43. Cohen's kappa values for word level perturbations per perturbation percentage using XLNet in the Sentiment Analysis task. The values above the bars show the proportion of perturbed samples for that given perturbation

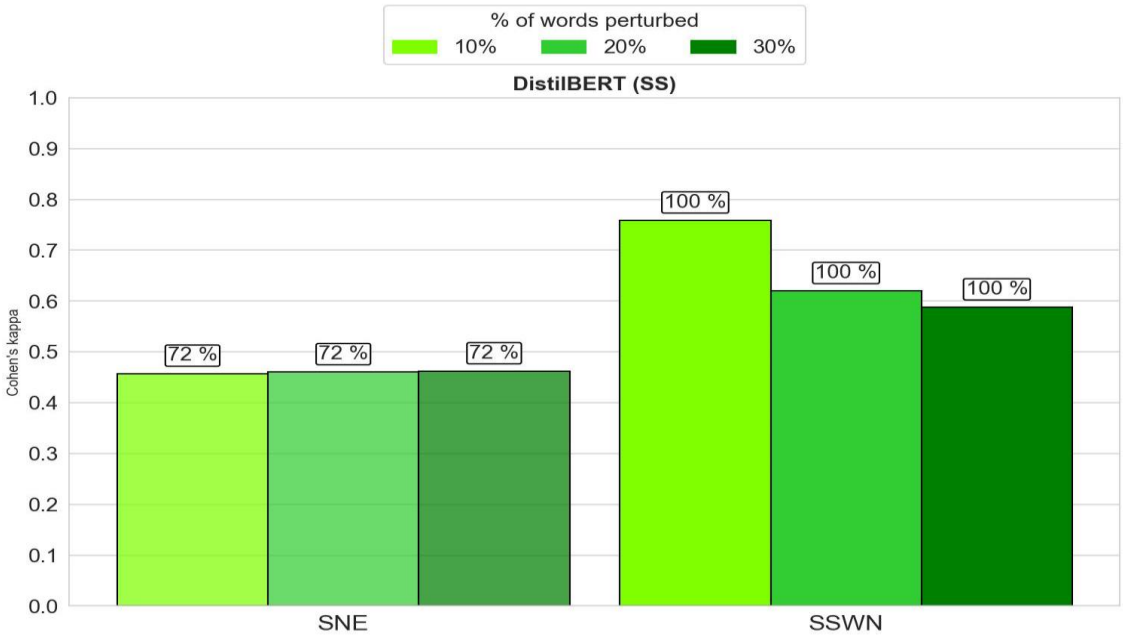

Figure 44. Cohen's kappa values for word level perturbations per perturbation percentage using DistilBERT in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

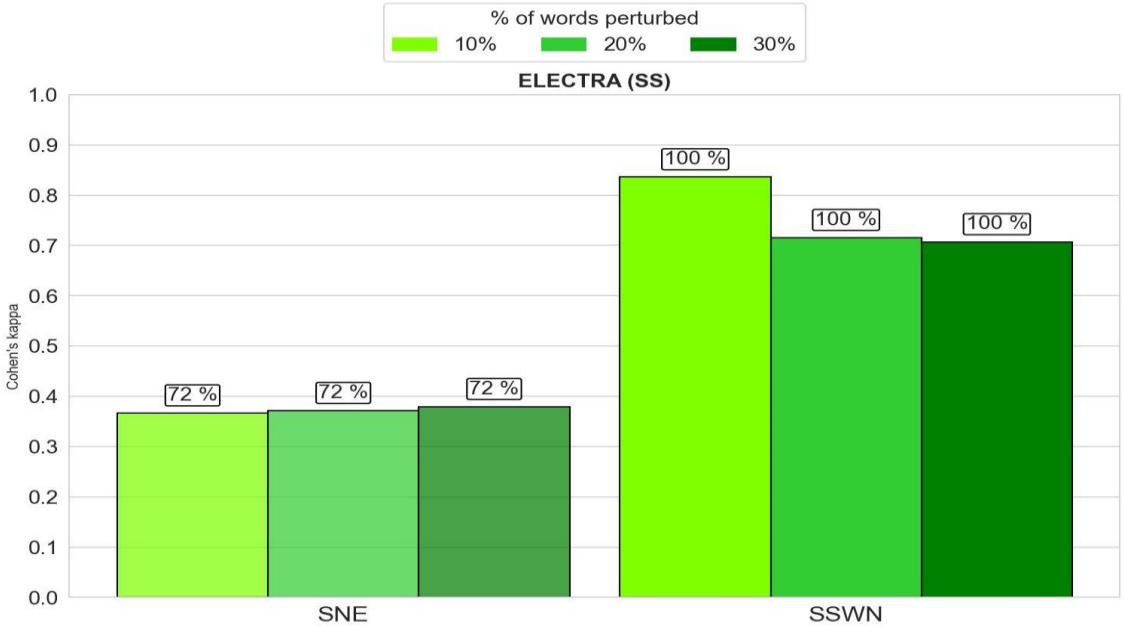

Figure 45. Cohen's kappa values for word level perturbations per perturbation percentage using ELECTRA in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

Figure 46. Cohen's kappa values for word level perturbations per perturbation percentage using Funnel Transformer in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

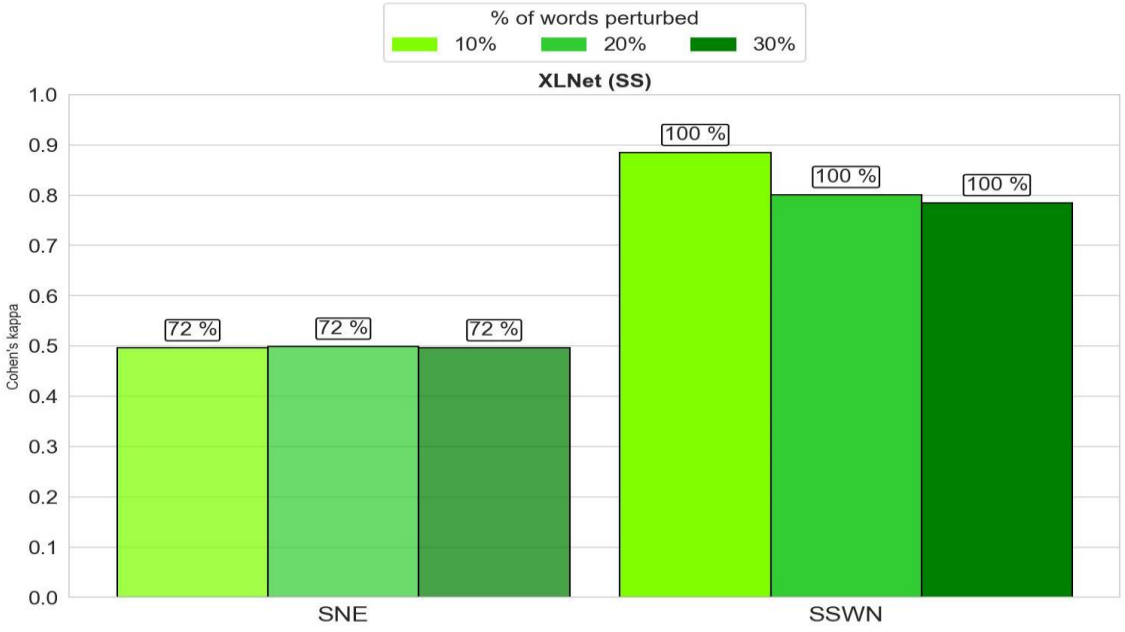

Figure 47. Cohen's kappa values for word level perturbations per perturbation percentage using XLNet in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

Other perturbations

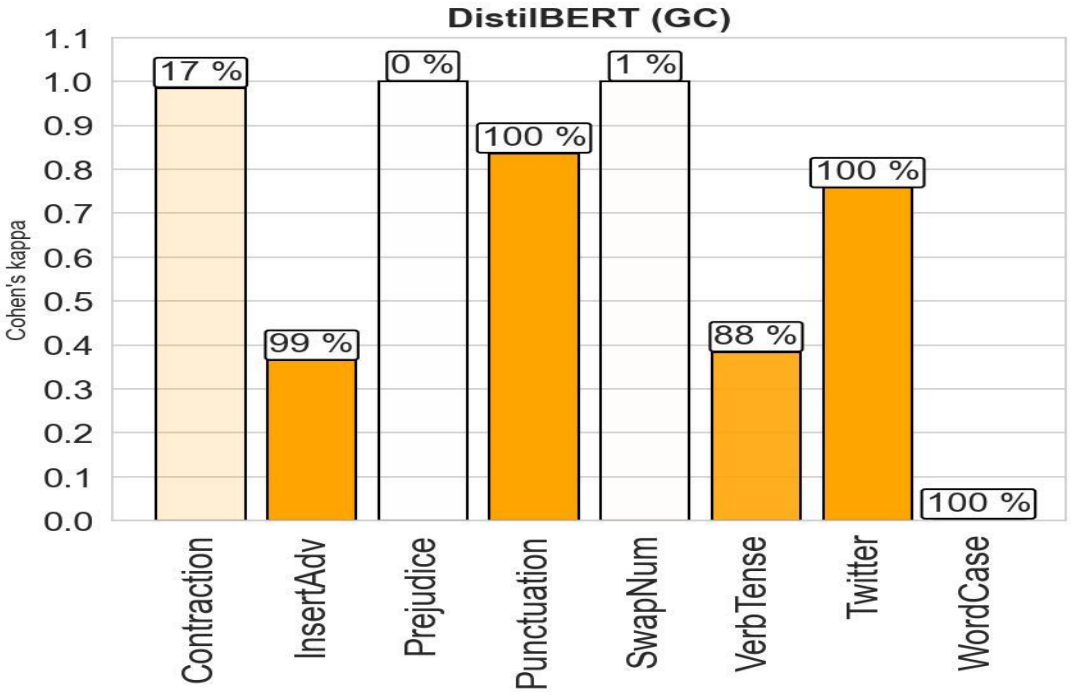

Figure 48. Cohen's kappa values for other perturbations per perturbation percentage using DistilBERT in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

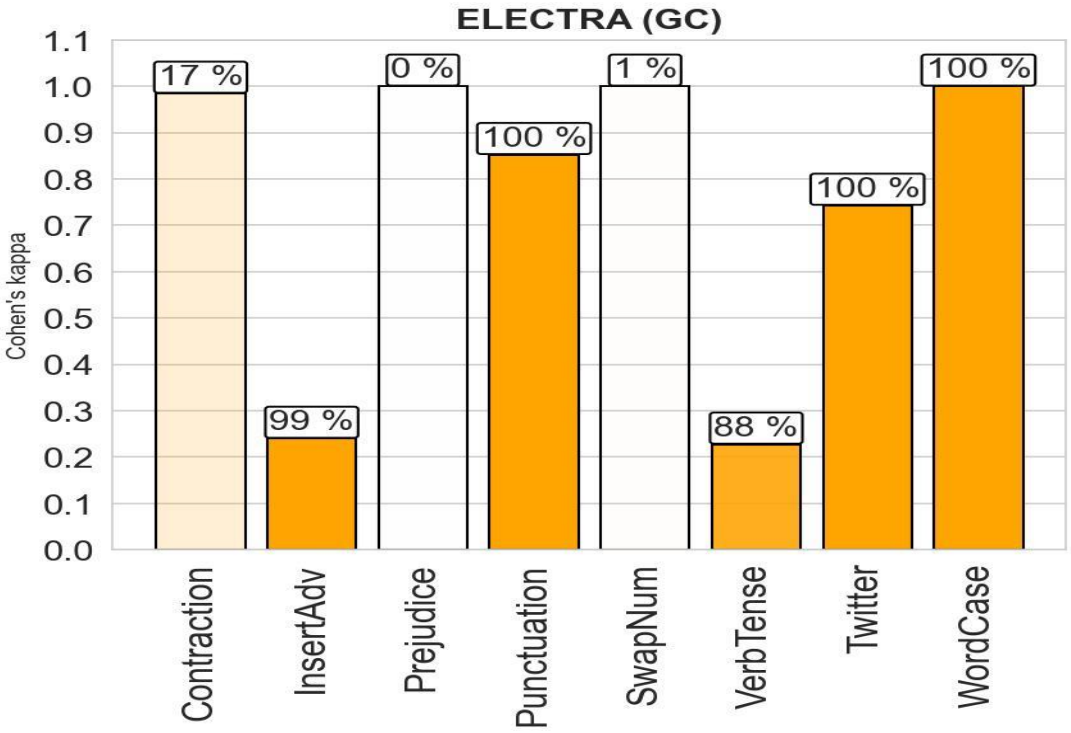

Figure 49. Cohen's kappa values for other perturbations per perturbation percentage using ELECTRA in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

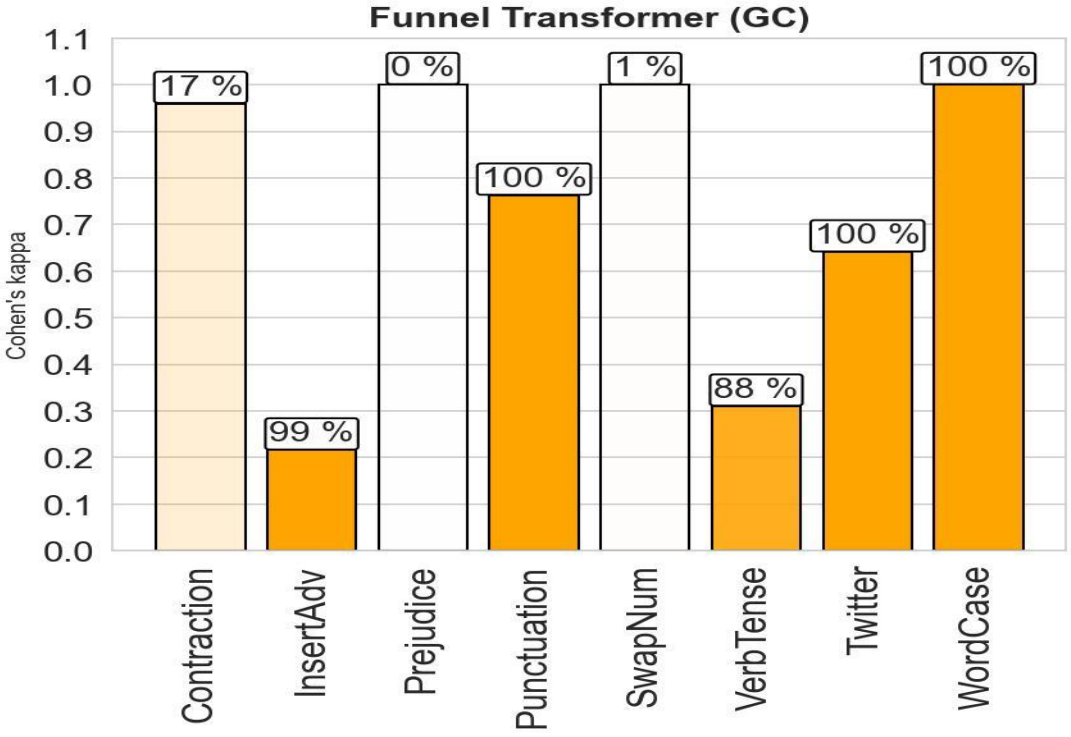

Figure 50. Cohen's kappa values for other perturbations per perturbation percentage using Funnel Transformer in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

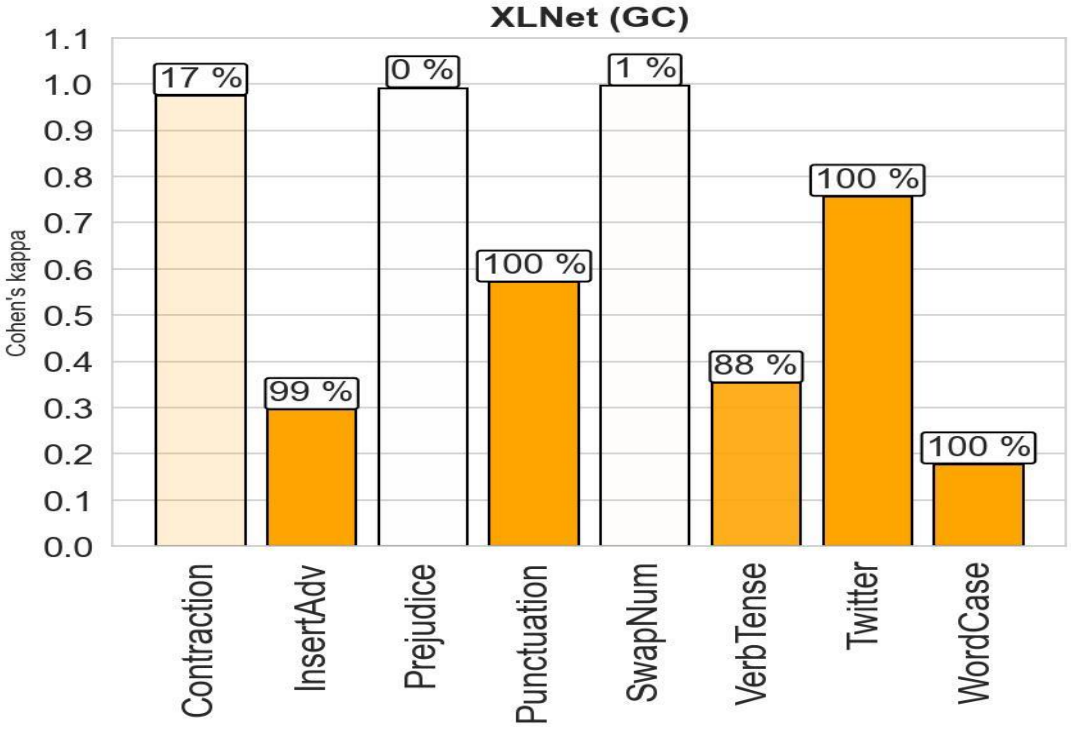

Figure 51. Cohen's kappa values for other perturbations per perturbation percentage using Funnel Transformer in the Grammatical Coherence task. The values above the bars show the proportion of perturbed samples for that given perturbation

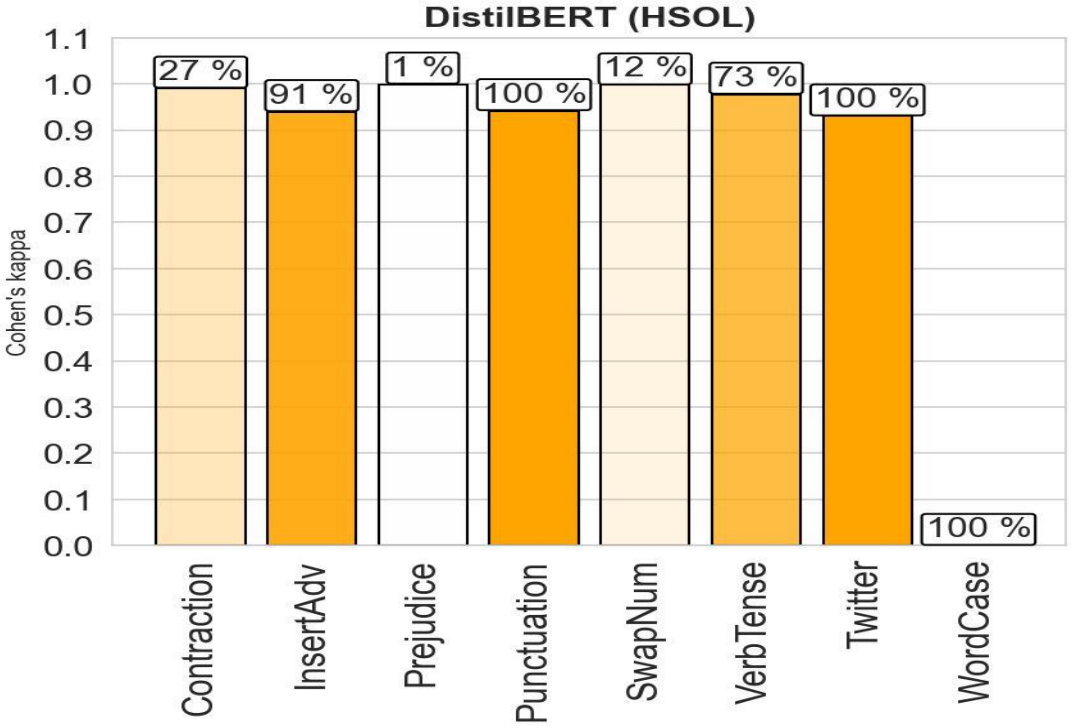

Figure 52. Cohen's kappa values for other perturbations per perturbation percentage using DistilBERT in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

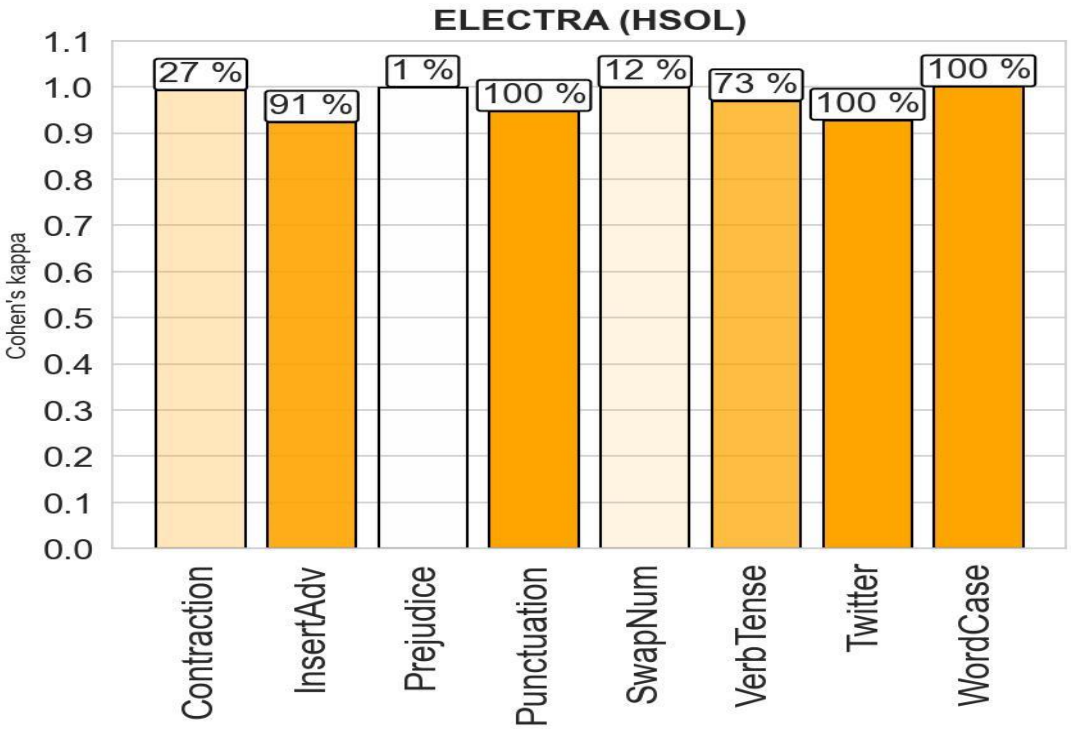

Figure 53. Cohen's kappa values for other perturbations per perturbation percentage using ELECTRA in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

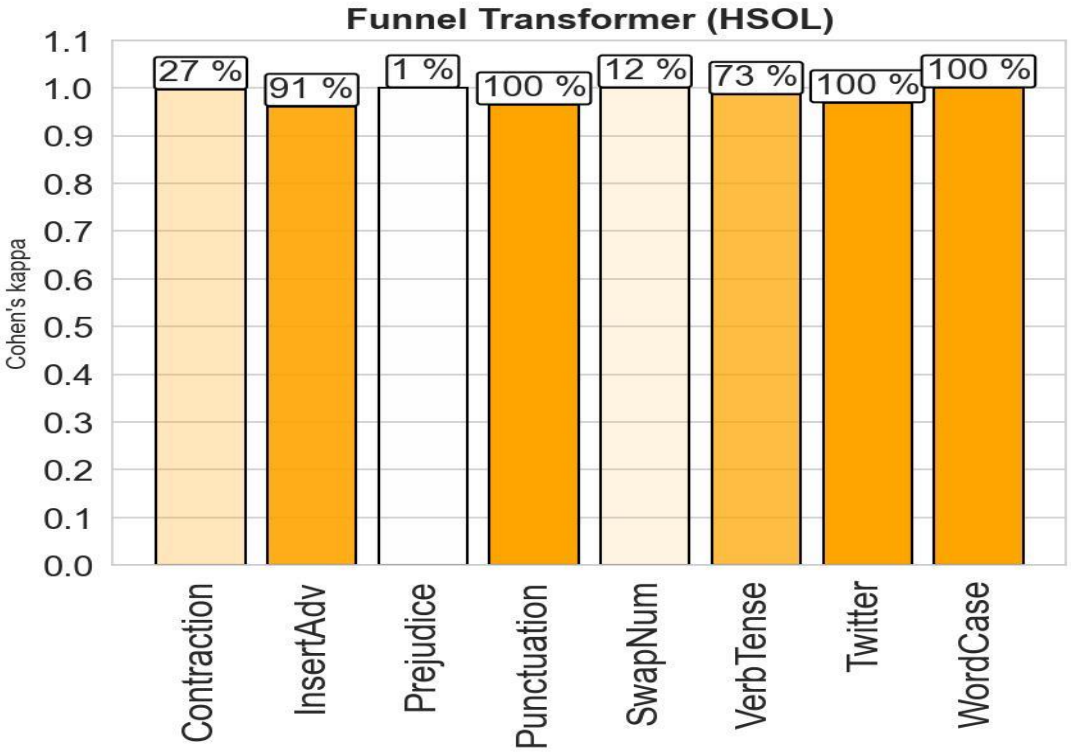

Figure 54. Cohen's kappa values for other perturbations per perturbation percentage using Funnel Tramsformer in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

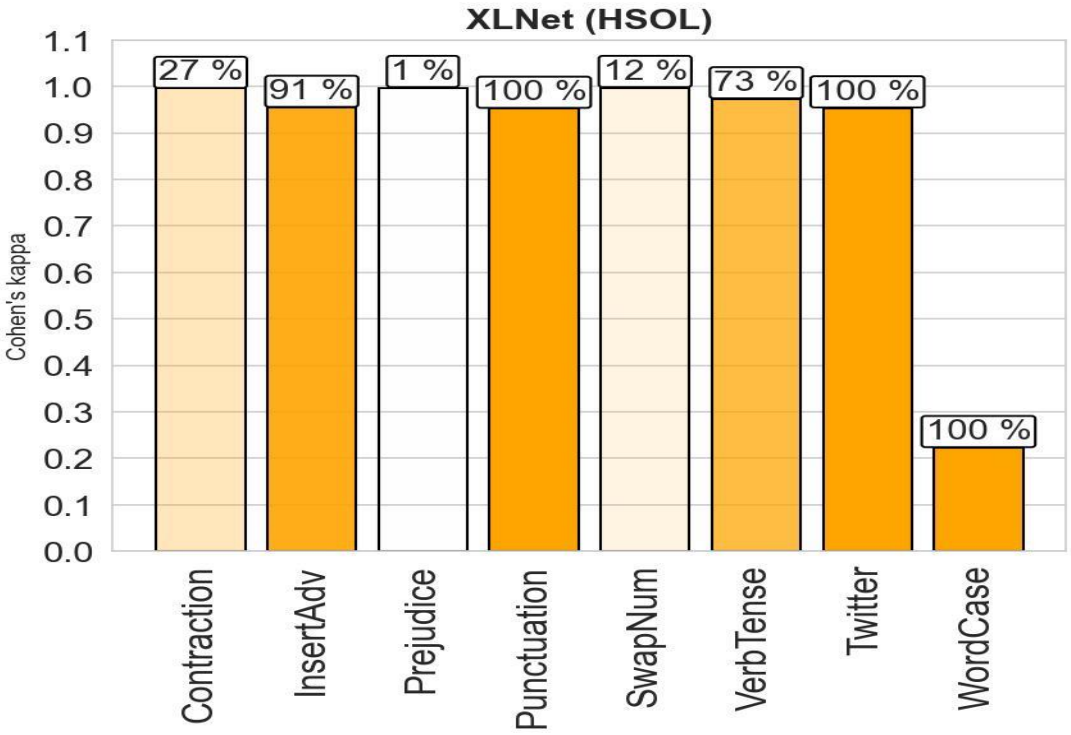

Figure 55. Cohen's kappa values for other perturbations per perturbation percentage using XLNet in the Hate Speech and Offensive Language task. The values above the bars show the proportion of perturbed samples for that given perturbation

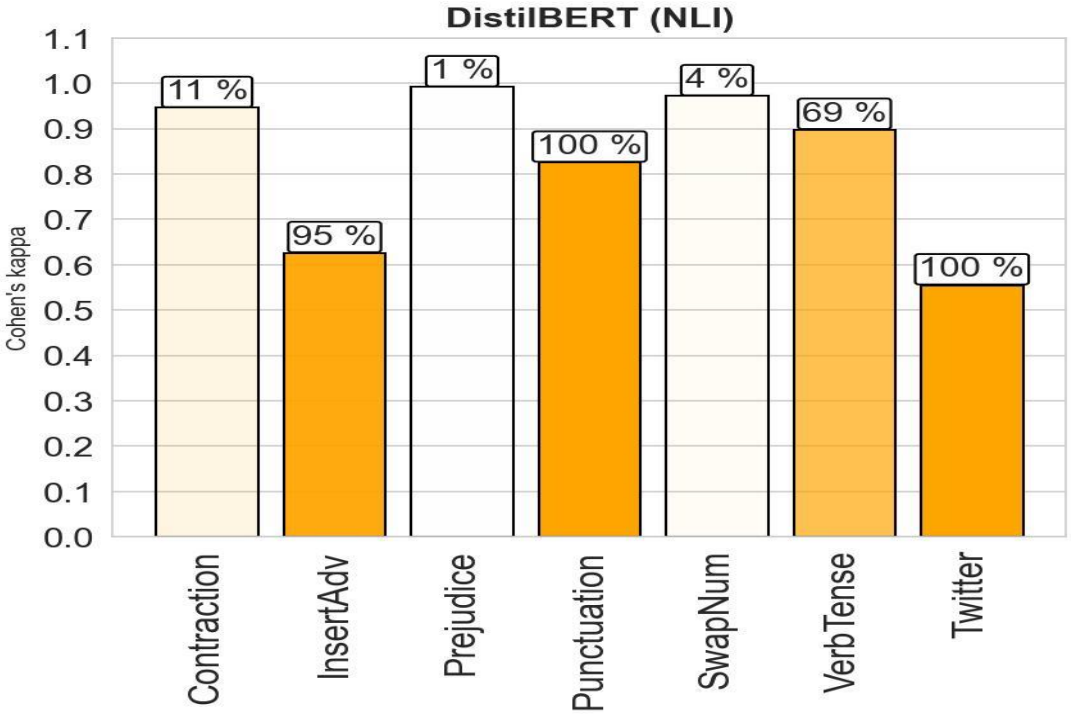

Figure 56. Cohen's kappa values for other perturbations per perturbation percentage using DistilBERT in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

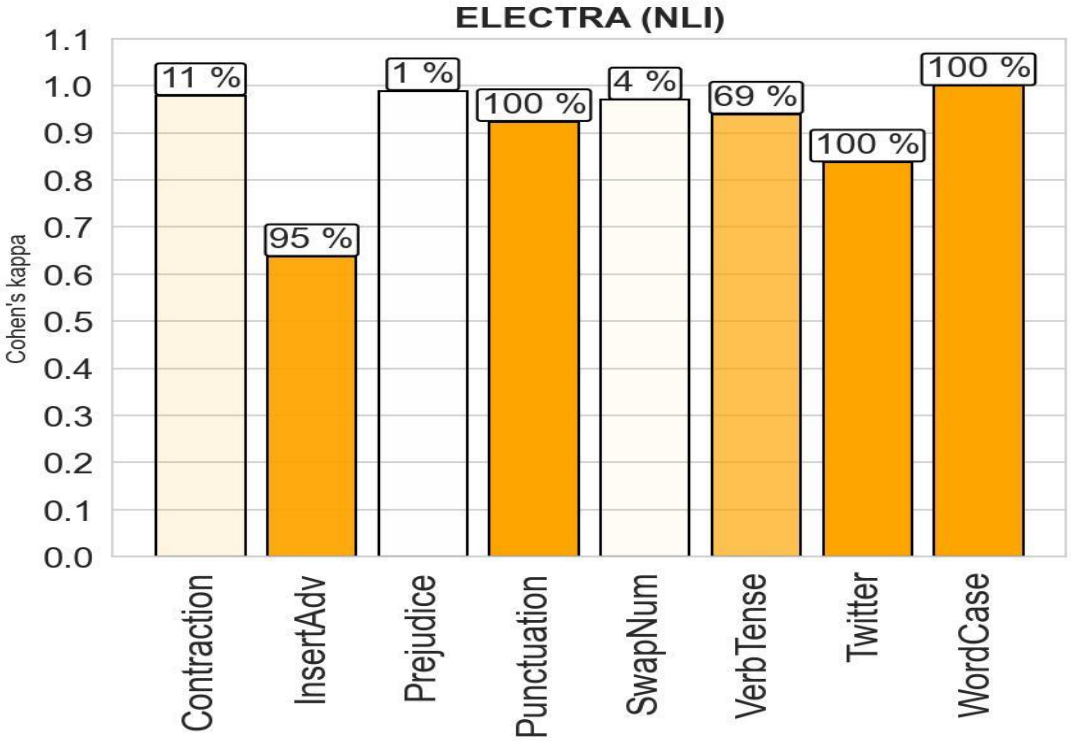

Figure 57. Cohen's kappa values for other perturbations per perturbation percentage using ELECTRA in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

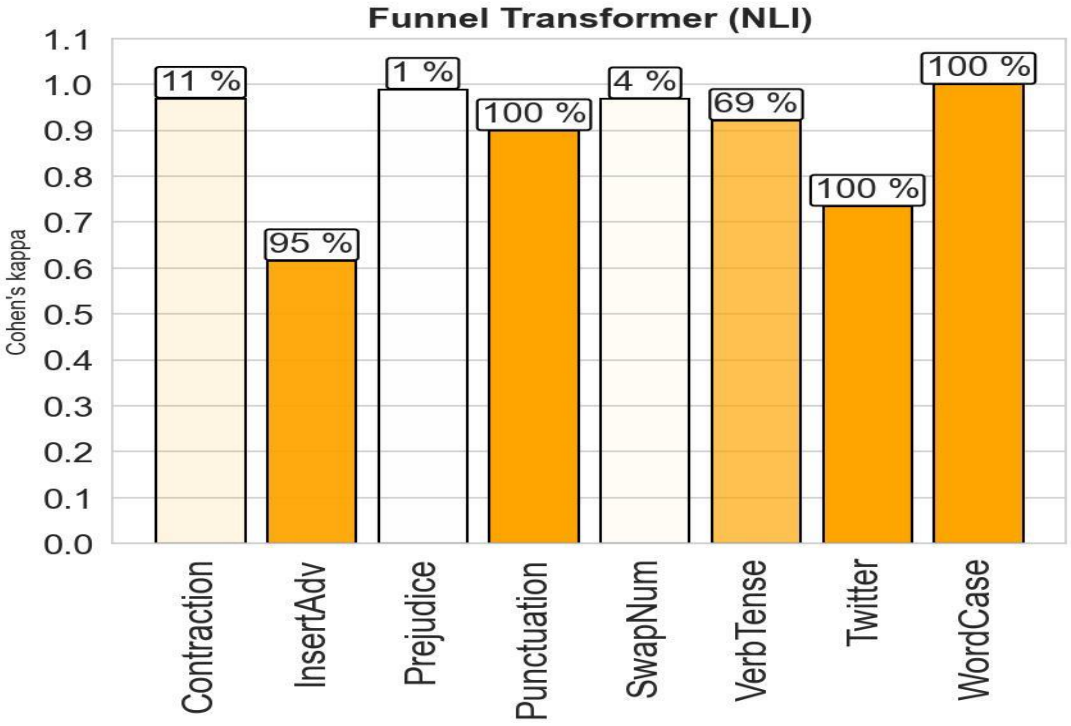

Figure 58. Cohen's kappa values for other perturbations per perturbation percentage using Funnel Transformer in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

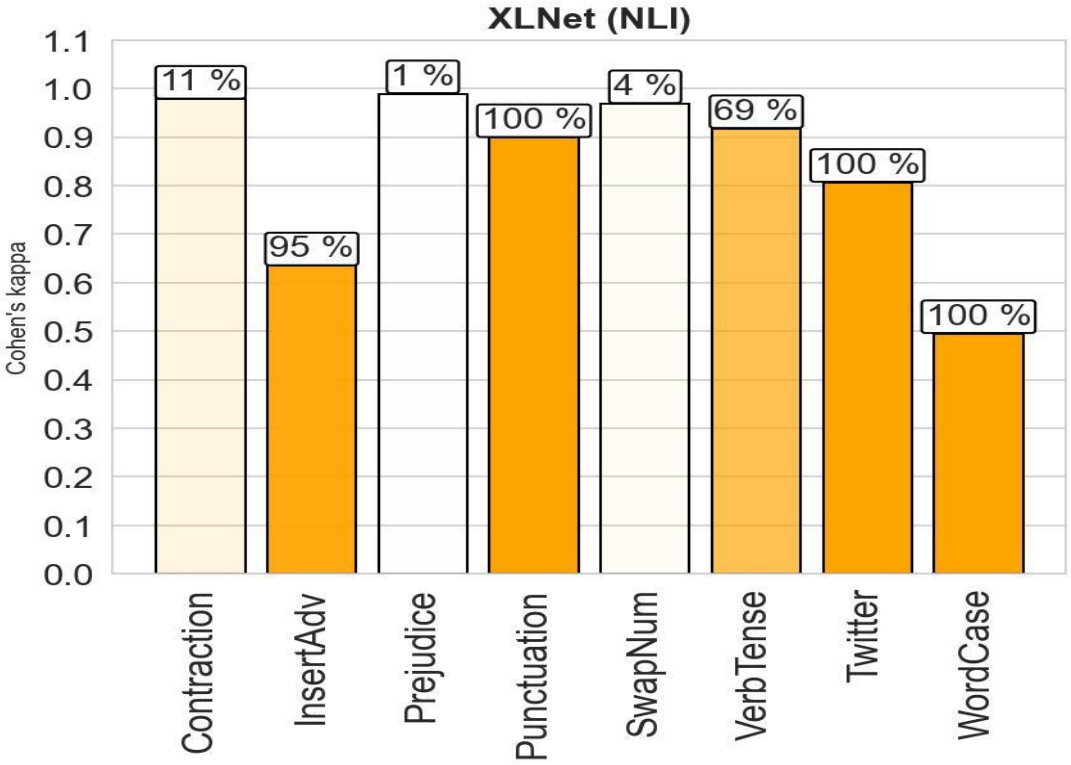

Figure 59. Cohen's kappa values for other perturbations per perturbation percentage using XLNet in the Natural Language Inference task. The values above the bars show the proportion of perturbed samples for that given perturbation

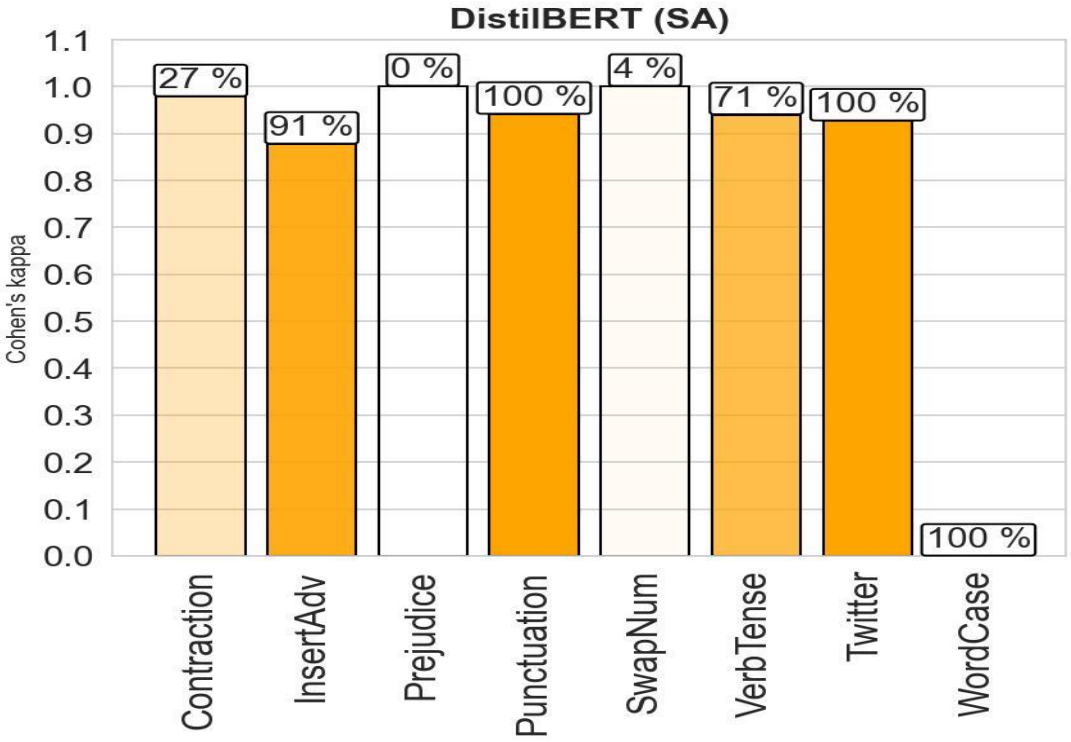

Figure 60. Cohen's kappa values for other perturbations per perturbation percentage using DistilBERT in the Sentiment Analysis task. The values above the bars show the proportion of perturbed samples for that given perturbation

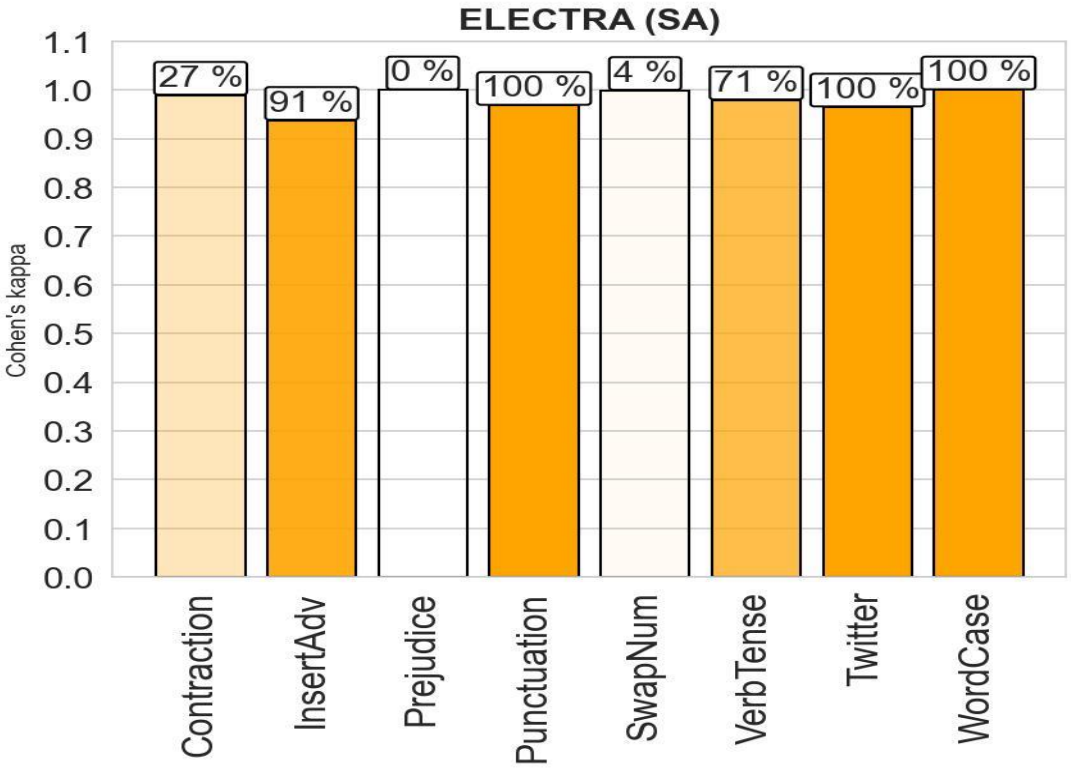

Figure 61. Cohen's kappa values for other perturbations per perturbation percentage using ELECTRA in the Sentiment Analysis task. The values above the bars show the proportion of perturbed samples for that given perturbation

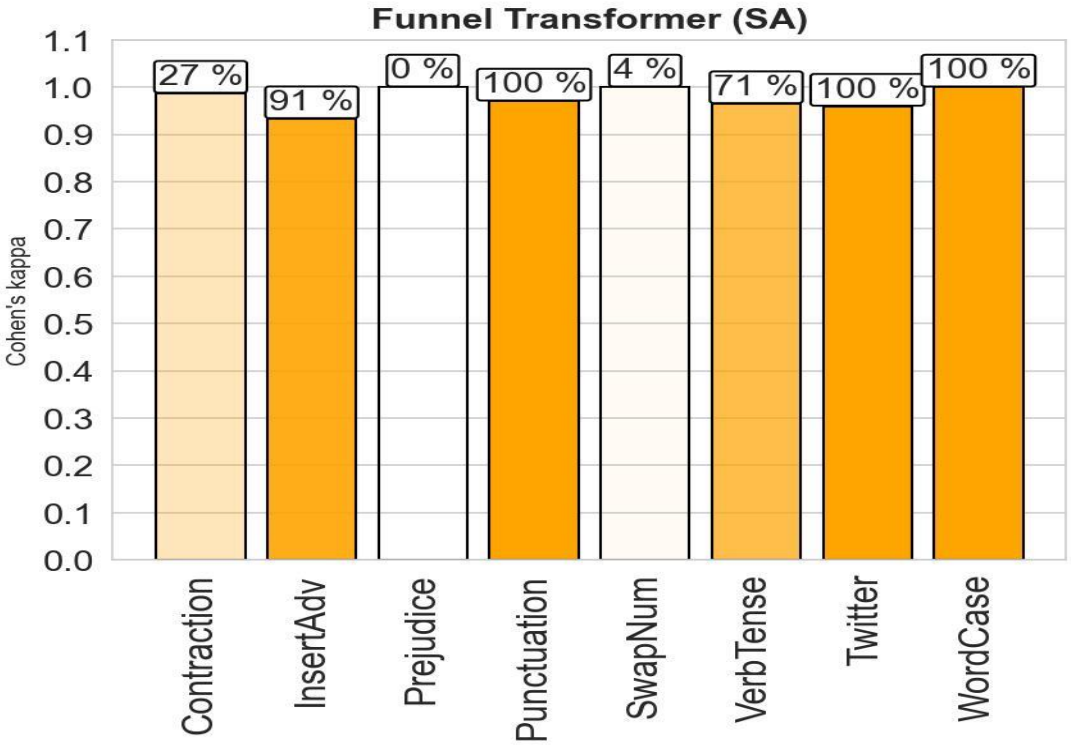

Figure 62. Cohen's kappa values for other perturbations per perturbation percentage using Funnel Transformer in the Sentiment Analysis task. The values above the bars show the proportion of perturbed samples for that given perturbation

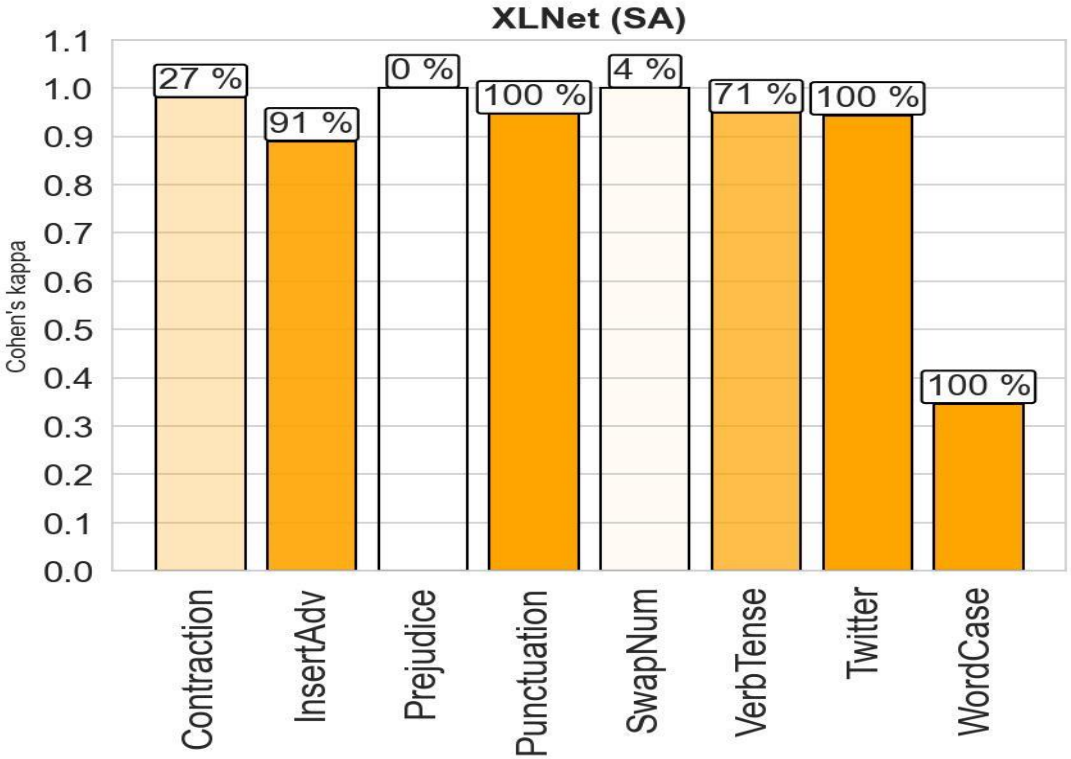

Figure 63. Cohen's kappa values for other perturbations per perturbation percentage using XLNet in the Sentiment Analysis task. The values above the bars show the proportion of perturbed samples for that given perturbation

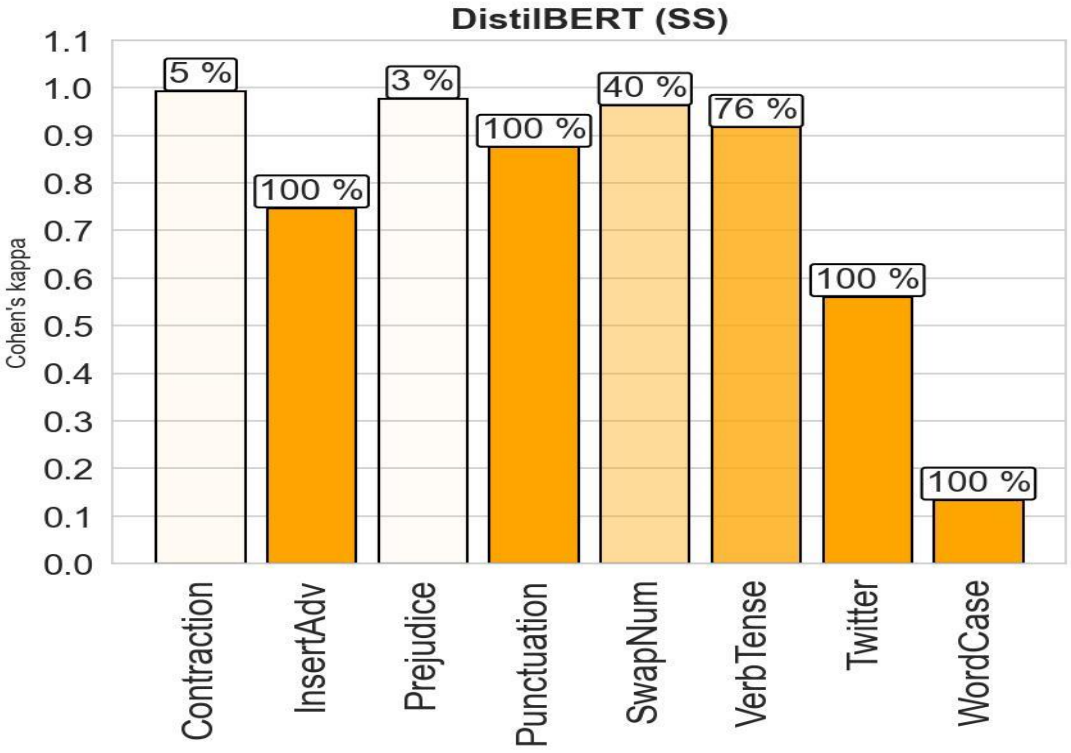

Figure 64. Cohen's kappa values for other perturbations per perturbation percentage using DistilBERT in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

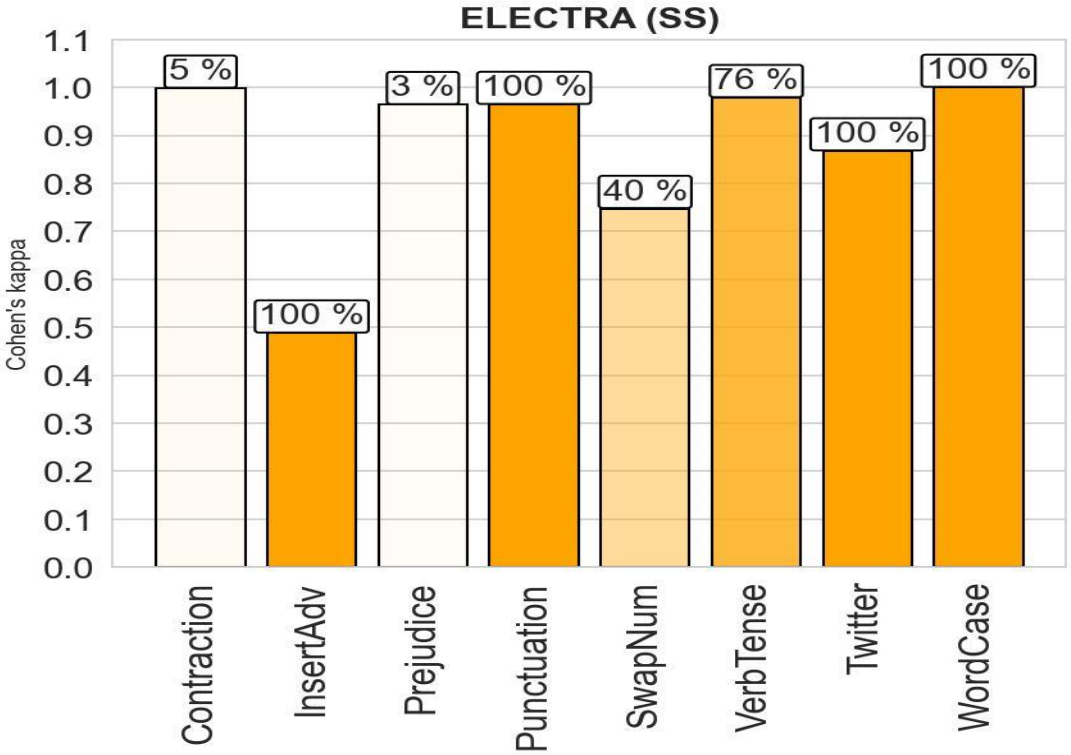

Figure 65. Cohen's kappa values for other perturbations per perturbation percentage using ELECTRA in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

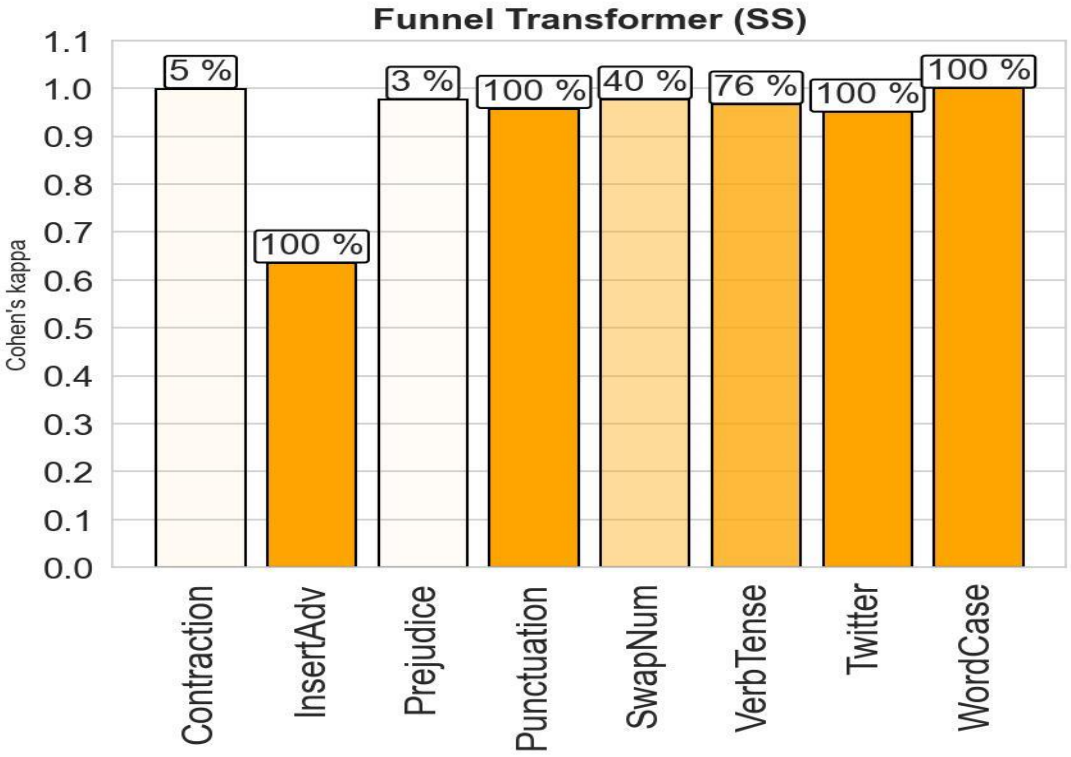

Figure 66. Cohen's kappa values for other perturbations per perturbation percentage using Funnel Transformer in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

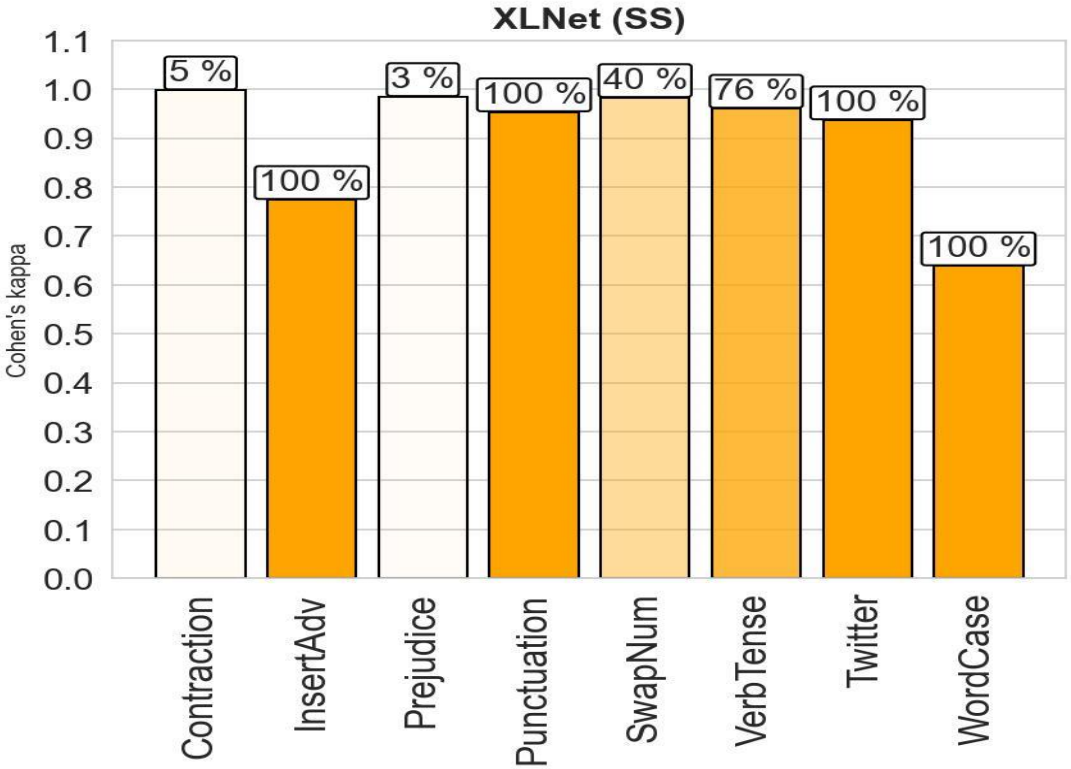

Figure 67. Cohen's kappa values for other perturbations per perturbation percentage using XLNet in the Semantic Similarity task. The values above the bars show the proportion of perturbed samples for that given perturbation

Tables

Methodology

Model	Based of	Pretraining approach / Architecture	Computation cost	Year
DistilBERT	BERT	Knowledge distilation	Cheap	2019
ELECTRA	BERT	Discrimnation based of generator-discriminator models rather than generation	Cheap	2020
XLNet	Transformer-XL	Permutation of tokens	Expensive	2020
Funnel Transformer	-	Reducing size at each block by pooling	Cheap	2020

Table 1. Summary of the characteristics of the four models employed: DistilBERT, ELECTRA, XLNet and Funnel Transformer

Sentence	Label
cold movie	0 (negative)
with his usual intelligence and subtlety	1 (positive)
will find little of interest in this film, which is often preachy and poorly acted	0 (negative)
a $ 40 million version of a game	0 (negative)
gorgeous and deceptively minimalist	1 (positive)

Table 2. Examples for the Stanford Sentiment Treebank with their corresponding label

Sentence	Label
As you eat the most, you want the least	0 (not acceptable)
John was lots more obnoxious than Fred	1 (acceptable)
The tube was escaped by gas	0 (not acceptable)
We want John to win	1 (acceptable)
We persuaded Mary to leave and Sue to stay	1 (acceptable)

Table 3. Examples for the Corpus of Linguistic Acceptability with their corresponding label

Sentence 1	Sentence 2	Label
It is the seat of Zerendi District in Akmola Region	It is the seat of the district of Zerendi in Akmola region	1 (Semantically similar)
BA transferred the former BCal routes to Heathrow to Tokyo and Saudi Arabia	BA relocated the former BCal routes to Tokyo and Saudi Arabia to Heathrow	0 (Not semantically similar)
He is trained by Andre Rozier and shares a gym with the former World Champion Daniel Jacobs	He is trained by Daniel Jacobs and shares a gym with former world champion Andre Rozier	0 (Not semantically similar)
The Tabaci River is a tributary of the River Leurda in Romania	The Leurda River is a tributary of the River Tabaci in Romania	0 (Not semantically similar)
Steam can also be used , and does not need to be pumped	Also steam can be used and need not be pumped	1 (Semantically similar)

Table 4. Examples of the Paraphrase Adversaries from Word Scrambling dataset with their corresponding label

Premise	Hypothesis	Label
Gays and lesbians	Heterosexuals	2 (Contradiction)
yeah i mean just when uh the they military paid for her education	The military didn't pay for her education	2 (Contradiction)
(Read for Slate's take on Jackson's findings)	Slate had an opinion on Jackson's findings	0 (Entailment)
yeah well you're a student right	Well you're a mechanics student right?	1 (Neutral)
Well you're a mechanics student right?	yeah well you're a student right	0 (Entailment)

Table 5. Examples of the Multi-Genre Natural Language Inference Corpus dataset with their corresponding label

Sentece	Label
@rhythmixx_ :hobbies include: fighting Mariam bitch	1 (Offensive langauge)
@AllAboutManFeet: http://t.co/3gzUpfuMev woof woof and hot soles	2 (Neither)
@CB_Baby24: @white_thunduh alsarabsss hes a beaner smh you can tell hes a mexican	0 (Hate Speech)
@CauseWereGuys: Going back to school sucks more dick than the hoes who attend it	1 (Offensive langauge)
@ArizonasFinest6: Why the eggplant emoji doe? y he say she looked like scream lmao	2 (Neither)

Table 6. Examples of the Hate Speech and Offensive Language dataset with their corresponding label

Dataset	NLP Task	Size	Nº labels
CoLA	Grammatical Acceptability	Train: 8 550 sentences	2
		Validation: 1 040 sentences	
		Test: 1 060 sentences	
HSOL	Hate Speech and Offensive Language	Train: 13 878 tweets	3
		Validation: 5 948 tweets	
		Test: 4 957 tweets	
MNLI	Natural Language Inference	Train: 15 000 pairs of senteces	3
		Validation: 9 820 pairs of sentences	
		Test: 9800 pairs of sentences	
SST2	Sentiment Analysis	Train: 15 000 sentences	2
		Validation: 872 sentences	
		Test: 1820 sentences	
PAWS	Semantic Similarity	Train: 15 000 pairs of senteces	2
		Validation: 8 000 pairs of sentences	
		Test: 8 000 paris of sentences	

Table 7. Summary of the five datasets used: CoLA, HSO, MNLI, SST2 and PAWS

Level	Perturbation	Original	Perturbed
Character	Keyboard	hand	hanf
	Ocr	hello	hellu
	SpellingError	achieve	acheive
	Typos	random	andom
Word	SwapSynWordEmbedding	It is nothing but a caricature	It is nothing but an imitation
	SwapSynWordNet	I'd like some pie, please	I'd like some cake, please
Other	Contraction	They are coming, run!	They're coming, run!
	InsertAdv	He jumped the fence	He quickly jumped the fence
	Prejudice	I'm moving to Canada next month	I'm moving to Morocco next month
	Punctuation	The film was quite good	The film was quite good!
	SwapNum	3 people attended the meeting	15 people attended the metting
	VerbTense	He started slow but ended up winning	He starts slow but ends up winning
	Twitter	Good premise, bad ending	Good premise, bad ending @username789
	WordCase	I'm not angry, not at all	I'M NOT ANGRY, NOT AT ALL

Table 8. Summary of the perturbations used with a short example

Finetuning

Hyperparameter	Value
Batch size	32 (16 XLNet)
Initial learning rate	0.0001 / 0.0005 / 0.00001
Learning rate decay	Linear decay per epoch
# of epochs	5 / 7 / 10
Optimiser	Adam with weight decay (AdamW)
Weight decay	0.01
Adam β_1	0.9
Adam β_2	0.999
Adam ε	1e-8

Table 9. Hyperparameters employed in the finetuning phase. The only two who take various values are the initial learning rate and the number of epochs

		Initial learning rate		
		0.00001	0.00005	0.0001
Number of epochs	5	0.874630	0.881330	0.871070
	7	0.878898	0.879571	0.881094
	10	0.879833	**0.884099**	0.874800

Table 10. Example of the scores obtained for the finetuning phase in the case of the DistilBERT model in the Grammatical Coherence task. The highest score (optimal combination of hyperparameters) is highlighted

		NLP Task									
		GC (CoLA)		HSOL (HSO)		NLI (MNLI Corpus)		SA (SST2)		SS (PAWS)	
Model	DistilBERT	10	0.00005	7	0.00005	7	0.00005	7	0.00005	10	0.00005
	ELECTRA	10	0.00005	10	0.00005	5	0.00005	5	0.00005	7	0.00005
	Funnel Transformer	10	0.00001	5	0.00001	7	0.00001	10	0.00001	10	0.00005
	XLNet	10	0.00001	7	0.00005	10	0.00001	5	0.00001	5	0.00005

Table 11. Best number of epochs (left) and initial learning rate (right) for each combination of model and NLP task

Detailed evaluation

Character level perturbations

Grammatical Coherence (CoLA) - Character level						
Perturbation	% of characters per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Keyboard	1%	0.41	**0.47**	0.37	0.43	100.0%
	5%	0.28	**0.30**	0.20	0.23	100.0%
	10%	0.08	**0.12**	0.05	0.06	100.0%
Ocr	1%	0.39	**0.44**	0.33	0.41	99.8%
	5%	0.21	**0.36**	0.25	0.29	99.8%
	10%	0.13	**0.17**	0.06	0.08	99.8%
SpellingError	1%	0.38	0.38	0.37	**0.39**	99.2%
	5%	**0.28**	0.26	0.18	0.27	99.2%
	10%	0.10	**0.10**	0.07	0.09	99.2%
Typos	1%	0.44	**0.48**	0.41	0.42	95.6%
	5%	0.32	**0.36**	0.24	0.32	97.6%
	10%	0.10	**0.16**	0.07	0.10	99.2%

Table 12. Scores for each model and character level perturbation type for the Grammatical Coherence task. The best score for each perturbation is highlighted

Hate Speech and Offensive Language (hate_speech_offensive) - Character level						
Perturbation	% of characters per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Keyboard	1%	0.81	0.82	**0.84**	0.83	100.0%
	5%	0.62	0.62	0.65	**0.66**	100.0%
	10%	0.35	0.33	0.35	**0.40**	100.0%
Ocr	1%	0.83	0.81	0.82	**0.86**	100.0%
	5%	0.66	0.59	0.60	**0.72**	100.0%
	10%	0.41	0.31	0.31	**0.50**	100.0%
SpellingError	1%	0.79	0.80	**0.82**	0.81	98.5%
	5%	0.59	0.61	**0.64**	**0.64**	98.6%
	10%	0.38	0.41	0.45	**0.46**	98.6%
Typos	1%	0.83	0.81	0.83	**0.86**	97.0%
	5%	0.60	0.60	0.61	**0.64**	99.4%
	10%	0.36	0.29	0.34	**0.39**	99.9%

Table 13. Scores for each model and character level perturbation type for the Hate Speech and Offensive Language task. The best score for each perturbation is highlighted

Natural Language Inference (MNLI Corpus) - Character level						
Perturbation	% of characters per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Keyboard	1%	0.63	0.74	0.73	**0.74**	99.9%
	5%	0.45	0.53	0.54	**0.57**	99.9%
	10%	0.18	0.22	0.29	**0.33**	99.9%
Ocr	1%	0.63	0.73	0.72	**0.73**	99.8%
	5%	0.44	0.52	0.53	**0.55**	99.8%
	10%	0.21	0.24	0.32	**0.34**	99.8%
SpellingError	1%	0.68	**0.77**	0.75	0.74	98.7%
	5%	0.49	0.59	**0.60**	0.60	98.8%
	10%	0.28	0.36	**0.40**	0.39	98.9%
Typos	1%	0.65	**0.75**	0.72	0.73	93.3%
	5%	0.46	0.55	0.54	**0.56**	98.2%
	10%	0.22	0.27	0.33	**0.35**	99.6%

Table 14. Scores for each model and character level perturbation type for the Natural Language Inference task. The best score for each perturbation is highlighted

Sentiment Analysis (Stanford Sentiment Treebank) - Character level						
Perturbation	% of characters per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Keyboard	1%	0.88	0.91	**0.93**	0.91	100.0%
	5%	0.69	0.75	**0.77**	0.77	100.0%
	10%	0.28	0.43	0.37	**0.47**	100.0%
Ocr	1%	0.86	**0.93**	0.93	0.92	100.0%
	5%	0.67	**0.77**	0.75	0.74	100.0%
	10%	0.28	0.45	0.34	**0.47**	100.0%
SpellingError	1%	0.91	0.94	**0.94**	0.92	99.9%
	5%	0.70	0.80	**0.81**	0.78	99.9%
	10%	0.52	0.62	0.63	**0.64**	99.9%
Typos	1%	0.90	0.92	**0.93**	0.89	99.9%
	5%	0.69	**0.76**	0.75	**0.76**	100.0%
	10%	0.32	0.45	0.32	**0.46**	99.9%

Table 15. Scores for each model and character level perturbation type for the Sentiment Analysis task. The best score for each perturbation is highlighted

Semantic Similarity (PAWS) - Character level						
Perturbation	% of characters per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Keyboard	1%	0.66	0.70	0.75	**0.85**	100.0%
	5%	0.13	0.08	0.22	**0.37**	100.0%
	10%	0.04	0.03	0.10	**0.17**	100.0%
Ocr	1%	0.69	0.65	0.77	**0.84**	100.0%
	5%	0.21	0.08	0.28	**0.38**	100.0%
	10%	0.14	0.09	0.30	**0.31**	100.0%
SpellingError	1%	0.73	0.77	0.82	**0.87**	100.0%
	5%	0.30	0.30	0.43	**0.55**	100.0%
	10%	0.24	0.29	0.42	**0.51**	100.0%
Typos	1%	0.67	0.70	0.79	**0.85**	100.0%
	5%	0.16	0.12	0.26	**0.44**	100.0%
	10%	0.07	0.06	0.20	**0.32**	100.0%

Table 16. Scores for each model and character level perturbation type for the Semantic Similarity task. The best score for each perturbation is highlighted

Word level perturbations

Grammatical Coherence (CoLA) - Word level						
Perturbation	% of words per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Swap Synonym Word Embedding	10%	0.89	**0.92**	0.87	0.87	39.7%
	20%	0.88	0.88	0.86	**0.89**	39.7%
	30%	0.88	0.89	0.88	**0.89**	39.7%
Swap Synonym WordNet	10%	**0.46**	0.32	0.32	0.37	99.8%
	20%	**0.39**	0.27	0.26	0.31	99.8%
	30%	**0.27**	0.16	0.16	0.15	99.8%

Table 17. Scores for each model and word level perturbation type for the Grammatical Coherence task. The best score for each perturbation is highlighted

Hate Speech and Offensive Language (hate_speech_offensive) - Word level						
Perturbation	% of words per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Swap Synonym Word Embedding	10%	0.97	0.97	**0.98**	0.97	31.1%
	20%	0.97	0.97	**0.98**	0.98	31.1%
	30%	0.98	0.97	**0.98**	0.97	31.1%
Swap Synonym WordNet	10%	0.69	0.75	**0.78**	0.71	99.1%
	20%	0.55	0.62	**0.67**	0.58	99.1%
	30%	0.42	0.51	**0.56**	0.46	99.1%

Table 18. Scores for each model and word level perturbation type for the Hate Speech and Offensive Language task. The best score for each perturbation is highlighted

Natural Language Inference (MNLI Corpus) - Word level						
Perturbation	% of words per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Swap Synonym Word Embedding	10%	**0.91**	0.87	0.86	0.87	19.3%
	20%	**0.91**	0.87	0.85	0.87	19.3%
	30%	**0.91**	0.87	0.85	0.87	19.3%
Swap Synonym WordNet	10%	0.70	**0.76**	0.74	0.75	99.3%
	20%	0.62	**0.70**	0.69	0.69	99.3%
	30%	0.54	**0.63**	0.61	0.63	99.3%

Table 19. Scores for each model and word level perturbation type for the Natural Language Inference Coherence task. The best score for each perturbation is highlighted

Sentiment Analysis (Stanford Sentiment Treebank) - Word level						
Perturbation	% of words per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Swap Synonym Word Embedding	10%	1.00	1.00	**1.00**	1.00	7.4%
	20%	1.00	0.99	**1.00**	1.00	7.4%
	30%	1.00	1.00	**1.00**	1.00	7.4%
Swap Synonym WordNet	10%	0.88	0.92	**0.93**	0.91	99.4%
	20%	0.81	0.88	**0.89**	0.86	99.4%
	30%	0.73	0.82	**0.83**	0.79	99.4%

Table 20. Scores for each model and word level perturbation type for the Sentiment Analysis task. The best score for each perturbation is highlighted

Semantic Similarity (PAWS) - Word level						
Perturbation	% of words per sample perturbed	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Swap Synonym Word Embedding	10%	0.46	0.37	0.44	**0.50**	72.1%
	20%	0.46	0.37	0.44	**0.50**	72.1%
	30%	0.46	0.38	0.45	**0.50**	72.1%
Swap Synonym WordNet	10%	0.76	0.84	0.86	**0.88**	100.0%
	20%	0.62	0.72	0.76	**0.80**	100.0%
	30%	0.59	0.71	0.74	**0.78**	100.0%

Table 21. Scores for each model and word level perturbation type for the Semantic Similarity task. The best score for each perturbation is highlighted

Other perturbations

Grammatical Coherence (CoLA) -Other					
Perturbation	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Contraction	**0.99**	0.98	0.96	0.98	16.7%
InsertAdv	**0.36**	0.24	0.22	0.30	99.4%
Prejudice	**1.00**	**1.00**	**1.00**	0.99	0.5%
Punctuation	0.84	**0.85**	0.76	0.57	100.0%
SwapNum	**1.00**	**1.00**	**1.00**	1.00	1.2%
VerbTense	**0.38**	0.23	0.31	0.35	88.1%
Twitter	**0.76**	0.74	0.64	0.76	100.0%
WordCase	0.00	**1.00**	**1.00**	0.18	100.0%

Table 22. Scores for each model and other perturbation type for the Grammatical Coherence task. The best score for each perturbation is highlighted

Hate Speech and Offensive Language (hate_speech_offensive) - Other					
Perturbation	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Contraction	0.99	0.99	**1.00**	1.00	26.5%
InsertAdv	0.94	0.92	**0.96**	0.95	90.6%
Prejudice	1.00	1.00	**1.00**	1.00	0.6%
Punctuation	0.94	0.95	**0.96**	0.95	100.0%
SwapNum	1.00	1.00	**1.00**	1.00	11.8%
VerbTense	0.98	0.97	**0.99**	0.97	73.4%
Twitter	0.93	0.93	**0.97**	0.95	100.0%
WordCase	0.00	**1.00**	**1.00**	0.22	100.0%

Table 23. Scores for each model and other perturbation type for the Hate Speech and Offensive Language task. The best score for each perturbation is highlighted

Natural Language Inference (MNLI Corpus) - Other					
Perturbation	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Contraction	0.95	**0.98**	0.97	0.98	11.0%
InsertAdv	0.63	**0.64**	0.62	0.63	94.7%
Prejudice	**0.99**	0.99	0.99	0.99	0.5%
Punctuation	0.83	**0.92**	0.90	0.90	100.0%
SwapNum	**0.97**	0.97	0.97	0.97	3.8%
VerbTense	0.90	**0.94**	0.92	0.92	68.7%
Twitter	0.56	**0.84**	0.73	0.81	100.0%
WordCase	0.35	**1.00**	**1.00**	0.49	99.9%

Table 24. Scores for each model and other perturbation type for the Natural Language Inference task. The best score for each perturbation is highlighted

Sentiment Analysis (Stanford Sentiment Treebank) - Other					
Perturbation	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Contraction	0.98	**0.99**	0.99	0.98	27.0%
InsertAdv	0.88	**0.94**	0.93	0.89	90.7%
Prejudice	**1.00**	**1.00**	**1.00**	**1.00**	0.0%
Punctuation	0.94	0.97	**0.97**	0.95	100.0%
SwapNum	**1.00**	1.00	**1.00**	**1.00**	4.1%
VerbTense	0.94	**0.98**	0.96	0.95	71.3%
Twitter	0.93	**0.96**	0.96	0.94	100.0%
WordCase	0.00	**1.00**	**1.00**	0.35	100.0%

Table 25. Scores for each model and other perturbation type for the Sentiment Analysis task. The best score for each perturbation is highlighted

Semantic Similarity (PAWS) - Other					
Perturbation	DistilBERT	ELECTRA	Funnel Transformer	XLNet	% of samples perturbed
Contraction	0.99	1.00	**1.00**	1.00	5.01%
InsertAdv	0.75	0.49	0.64	**0.77**	99.75%
Prejudice	0.98	0.96	0.98	**0.98**	2.86%
Punctuation	0.87	**0.96**	0.96	0.95	100.00%
SwapNum	0.96	0.75	0.98	**0.98**	40.15%
VerbTense	0.92	**0.98**	0.97	0.96	76.41%
Twitter	0.56	0.87	**0.95**	0.94	100.00%
WordCase	0.13	**1.00**	**1.00**	0.64	100.00%

Table 26. Scores for each model and other perturbation type for the Semantic Similarity task. The best score for each perturbation is highlighted

Aggregated evaluation

Model	Score
XLNet	0.487
Funnel Transformer	0.472
ELECTRA	0.464
DistilBERT	0.445

Table 27. Average kappa score per model

NLP Task	Score
SA	0.564
HSOL	0.536
SS	0.487
NLI	0.451
GC	0.297

Table 28. Average kappa score per NLP task

Perturbation type	Score
CHARACTER	0.495
OTHER	0.468
WORD	0.438

Table 29. Average kappa score per perturbation type

Chracter perturbation type	Score
SpellingError	0.535
Ocr	0.488
Typos	0.483
Keyboard	0.474

Table 30. Average kappa score per character perturbation

Word perturbation type	Score
SSWN	0.632
SNE	0.243

Table 31. Average kappa score per word perturbation. Values in red indicate that that the mean proportion of perturbed samples in the dataset is less than 50%

Other perturbation type	Score
Punctuation	0.898
Twitter	0.836
InsertAdv	0.642
WordCase	0.632
VerbTense	0.608
Contraction	0.169
SwapNum	0.115
Prejudice	0.009

Table 32. Average kappa score per other type of perturbation. Values in red indicate that that the mean proportion of perturbed samples in the dataset is less than 50%

Perturbation type	Score
OTHER	0.723
WORD	0.632
CHARACTER	0.495

Table 33. Average kappa score per perturbation type when non-represented perturbations are removed