Electra model for Sexism Categorization[*]

Mattia Patruno[1,*]

[1]*University of Bari "Aldo Moro", Umberto I Square - 70121 Bari (Italy)*

Abstract

In this project, we address the challenge of text categorization focusing on the identification of sexist content and the categorization of sexism types for the EDOS competition (Explainable Detection of Online Sexism) [1]. Leveraging the power of transformer-based models, we implemented a comprehensive pipeline that includes data preprocessing, augmentation, and classification using the ELECTRA model [2].

The ELECTRA model, fine-tuned on our augmented dataset, demonstrated significant improvements in identifying sexist content and categorizing different types of sexism, with respect to the BERT model.

Our results indicate that combining data augmentation strategy with transformer-based models significantly enhances classification accuracy with unbalanced real-world datasets. The final ELECTRA model achieved an F1 score of 0.804 on the test set, demonstrating its effectiveness in real-world applications.

Keywords

EDOS, Sexism Categorization, Text Categorization, ELECTRA model, BERT model, SVM

1. Introduction and Motivations

As indicated in the EDOS paper [1], online sexism is a widespread and harmful phenomenon. Automated tools can assist the detection of sexism at scale. Detecting and categorizing such content is essential for maintaining safer and more inclusive digital spaces. Recent advancements in Natural Language Processing (NLP) and transformer-based models like ELECTRA [2] offer powerful tools for tackling these challenges.

This project leverages the ELECTRA model to address sexism detection and categorization in online texts. We choose this model for its speed, reduced use of resources and above all precision, as it has a comparable effectiveness to the more costly BERT model [10].

Our approach includes a preprocessing pipeline, with lemmatization, stopword and punctuation removal, tokenization, augmentation through synonym replacement, and evaluation using K-fold cross-validation. These steps aim to improve the model's robustness and generalization. We compare the results of the chosen model with the results of a baseline created with a simple SVM (Support Vector Machine) commonly used for text categorization tasks due to its efficiency. However, as we will see, this efficiency can sometimes come at the cost of effectiveness in specific scenarios. Additionally, we compare the results of the ELECTRA model with those of the BERT model.

1.1. Task Description

In this project, we focus on **Task B - Category of Sexism** of the EDOS competition [1]. This task involves classifying sexist posts into one of four categories:

1. **Threats:** Posts that contain threats, plans to harm, or incitement.
2. **Derogation:** Posts that belittle or demean individuals or groups.
3. **Animosity:** Posts that express hostility or strong dislike.
4. **Prejudiced Discussions:** Posts that engage in biased or prejudiced discussions.

The goal is to develop a system capable of accurately predicting the category of sexism for each post already identified as sexist. For this reason, we don't focus on the sexism recognition part (binary classification).

NLP Course 2023-2024

[*] EDOS Task B: Category of Sexism

[*] Corresponding author.

2. Related Work

Text classification stands as a foundational pillar within natural language processing (NLP), serving as the bedrock for various applications that involve understanding and organizing textual data. At its core, text classification involves the automatic categorization of text documents into predefined classes or categories based on their content.[10]

In Traditional NLP, the classification process relied on handcrafted features and machine learning algorithms, which often struggle to capture the complexity and nuances of language, especially with large and unbalanced datasets.

Language models like BERT leverage transformer-based architectures to grasp the complexities of natural language more comprehensively.

BERT (and its derivative versions such as DistilBert[9]) stands out due to its bidirectional nature, enabling it to consider the full context of a word by analyzing both its preceding and subsequent words in a sequence.[10]

However, there are several naive approach to this problem, one of which is the use of Support Vector Machine for classification, less resource-costly than BERT model.

Support Vector Machines (SVMs) are powerful and versatile machine learning algorithms capable of performing linear and non-linear classification, regression, and outlier detection. For text categorization, SVMs are often used due to their effectiveness in high-dimensional spaces such as text data[5].

The process typically begins with text preprocessing, where all text is normalized and cleaned. The text is then tokenized, splitting it into individual words, and optionally, lemmatization is applied to reduce words to their base forms. We will adapt these preprocessing techniques for our approach.

Feature extraction follows the data preprocessing, utilizing techniques like Term Frequency-Inverse Document Frequency (TF-IDF) to convert the text into numerical feature vectors. In TF-IDF, each word's frequency is adjusted by its importance across all documents, thereby reducing the weight of common words.

For model training, the data is split into training and testing sets. The classifier, typically with a linear kernel, is trained using the training set. Prediction and evaluation involve using the trained model to predict the categories of the text in the test set, and the performance is evaluated using metrics such as accuracy, precision, recall, and F1 score.

As mentioned in section 3.1.1 and shown in section 4.1, the dataset is not well suitable for a naive classifier. For this reason, one of the solutions that best mitigates the problem is the use of a Language Model [8], or self-supervised language representation learning, such that the ELECTRA model [3] [2].

3. Proposed Approach

Our proposed approach involves leveraging the powerful ELECTRA model, fine-tuned for the specific task of sexism categorization. We will adopt some preprocessing tecniques from the naive SVM approach and use them to have a common ground useful for evaluating purposes. Then we will perform the task with a SVM, the BERT model and ELECTRA model and compare the results.

3.0.1. Selection of the ELECTRA Model

As indicated in the study [2] *"Masked language modeling (MLM) pre-training methods such as BERT corrupt the input by replacing some tokens with [MASK] and then train a model to reconstruct the original tokens. While they produce good results when transferred to downstream NLP tasks, they generally require large amounts of compute to be effective."*

So, we chose as main model the ELECTRA model requiring significant less computational power and memory to perform efficiently with respect to the BERT model. ELECTRA's pre-training method, which involves training a discriminator to distinguish real input tokens from replaced tokens, has been shown to be more compute-efficient and achieve better downstream task performance than BERT [2] [7]. ELECTRA models have consistently outperformed BERT models in benchmarks, making them a suitable choice for our task.

3.0.2. Data Augmentation with Synonym Replacement

To augment the data, we employed synonym replacement, which involves replacing words in the text with their synonyms. This technique increases the diversity of the training data without altering the original meaning, helping the model generalize better. Previous research has shown that data augmentation techniques, such as synonym replacement, can improve model robustness and performance in text classification tasks [4]. By introducing variability in the training data, we aim to mitigate overfitting and enhance the model's ability to handle diverse expressions of sexism.

3.0.3. Data Preprocessing

As mentioned in the introduction of section 3 and section 2 we will use a preprocessing pipeline with:

1. Text normalization
2. Text cleaning (stopwords and punctuation removal)
3. Lemmatization
4. Tokenization

A comprehensive text preprocessing pipeline begins with text normalization, where the text is converted to lowercase to ensure uniformity and remove case sensitivity. Following normalization, stopwords and punctuation are removed to eliminate common words that do not contribute significant meaning and to strip away extraneous symbols that do not affect the semantics of the text. The next step used is lemmatization, which involves reducing words to their base or root form, thus standardizing words like "running," "ran," and "runs" to "run," enhancing consistency and reducing dimensionality. Finally, tokenization is performed, splitting the text into individual words or tokens. A token could not be a single word, but a subsection of a word (like syllabs) or a set of subsection of words. This granularity is chosen by the pre-trained tokenizer, different from model to model.

Together, these steps ensure that the text data is clean, consistent, and ready for effective feature extraction and subsequent modeling. We have adopted those steps to each run with different model, ensuring a common ground for evaluation.

3.0.4. K-Fold Cross-Validation

We implemented K-fold cross-validation to ensure a robust evaluation of our model. This technique involves partitioning the dataset into K subsets, training the model on K-1 subsets, and validating it on the remaining subset. This process is repeated K times, with each subset used exactly once as the validation data. K-fold cross-validation provides a comprehensive assessment of model performance by reducing the variance associated with a single train-test split.[6].

3.1. Description of the solution and dataset

The dataset used is the official dataset for the competition, comprising text samples labeled for sexism and sexism categories. The dataset is strongly unbalanced, biased towards the category 2 Derogation. This is evident from the results of the chosen baseline as the SVM has results that orbit around category 2 (See section 4.1).

For this reason the solution involves the use of a LM for the classification task (See section 3.2 for futher details).

3.1.1. Dataset

This task has two big problems related to its dataset:

1. The categories all have the **same polarity**, so it is difficult to distinguish them clearly.
2. The dataset is **strongly biased** towards one category.

These two characteristics of the dataset and of the task itself tend to confuse the most trivial and naive classifiers. The use of an LM such as ELECTRA or BERT is essential to address those difficulties[8].

The official dataset for the EDOS competition [1] (edos labeled aggregated) has the following features:

- *rewire id*: the id of the contributor that insert this row in the dataset.
- *text*: the text object to the classification task.
- *label sexist*: a binary classification of sexist or not sexist text (TASK A of EDOS).
- *label category*: a 4 class classification of sexist text (TASK B of EDOS).
- *label vector*: a 11 class classification of sexist text (TASK C of EDOS).
- *split*: train or validation sets

This dataset is composed by 20000 entries, but only 4854 have a positive label on the field *"label sexist"*.

Also We have created a simple test dataset with the same structure and with 5 phrases for each category, for evaluation purposes (See section 3.3 to locate those files in the related github repository).

3.2. Main technical details

The main language chosen for this task is Python given the vast amount of libraries available for the text classification task. The libraries used in this project are:

- **Pandas** is a data manipulation and analysis library, used for loading and preprocessing the dataset.
- **NumPy** is a package for scientific computing in Python.
- **Scikit-learn** is a machine learning library, used for splitting the dataset, computing evaluation metrics like F1 score, precision, and recall, performing grid search for hyperparameter tuning, and training SVM classifiers.
- **Matplotlib** is a plotting library, used for visualizing the confusion matrix with the help of the library **Seaborn** for personalization purposes.
- **Transformers (Hugging Face)** is a library that provides state-of-the-art machine learning models for natural language processing, including BERT, ELECTRA, and more. It is used for tokenization of text data, loading pre-trained models and fine-tuning these models for the text categorization task.
- **NLTK (Natural Language Toolkit)** is a library for natural language processing, used for lemmatization and stopword removal tasks.
- **WordNet** is a lexical database for the English language, used to provide word synonyms for the synonym replacement data augmentation technique in combination with the library **TextAugment** to fulfill the task.
- **Torch (PyTorch)** is a machine learning library, that handles tensors and integrates with the **Transformers** library.
- **Accelerate (Hugging Face)** is a library that provides utilities to enable fast and efficient training and evaluation of machine learning models.
- **Evaluate (Hugging Face)** is used for evaluation of machine learning models.

The environment used for the execution of the Python scripts (.ipynb notebook format) is Google Collab (See section 3.3 for further details). The execution is done with the free subscription, so many performance evaluation may vary depending on the load of the virtual machines made available by Google. Furthermore, this theoretically prevents execution in the background, so we must constantly check our active presence on the site.

3.2.1. Implementation of the Support Vector Machine (SVM)

The first provided solution using a Support Vector Machine (SVM) with k-fold cross-validation is the baseline for our evaluation. We define a preprocessing pipeline that converts text to lowercase, removes punctuation and stopwords, and applies lemmatization. Then we loads and preprocesses the datasets, focusing on sexist texts and encoding their labels. The texts are split into training, validation, and test sets, which are further preprocessed with the augmentation task by synonym replacement. A k-fold cross-validation (with 3 splits) is performed to train and evaluate the SVM model, using TF-IDF vectorization within a SMV model pipeline. The performance metrics (accuracy, precision, recall, and F1 score) across the folds are computed. After cross-validation, the model is evaluated on the test set, with results including a confusion matrix visualization.

3.2.2. Implementation of BERT model

For the BERT model is applied the same preprocessing pipeline of the SVM. Then we instantiate the tokenizer and the model itself, in this case "bert-base-uncased". The tokenizer applies padding and truncation to a max length of 128 characters, for hardware limitations. For the same reason, the trainer of the model has a small batch size for train and evaluation tasks. We also use a standard learning rate ($5e^{-5}$) and 100 warmup steps (from 0 to the learning rate). A low weight decay value encourages the model to keep the weights small, thereby reducing the complexity of the model and improving generalization to new, unseen data[2]. The epochs are set to 1 for hardware and time limitations (See section 4.2). A k-fold cross-validation (with 3 splits) is performed to train and evaluate the BERT model. The performance metrics (accuracy, precision, recall, and F1 score) are computed for each folds. After cross-validation, the model is evaluated, predicting the labels of the test sets, with results including a confusion matrix visualization.

3.2.3. Implementation of ELECTRA model

For the ELECTRA model is applied the same process described in section 3.2.2, but because it necessitates of substantial less hardware resources and processing time wrt BERT model, the train epochs are set to 5 instead of only 1. Also, we use the tokenizer "electra-small-discriminator".

3.3. Other information useful to replicate the approach

We recommend the use of Google Collab (https://colab.research.google.com/) as enviroment for the execution of the implementations of the classifiers. For the maximum possible fidelity on the performance using the Google Collab free plan is recommended.

The implementations are stored in a github repository accessible at this address https://github.com/m3ttiw/nlp-exam-edos-project. In the folder *datasets* are stored the two dataset used in this project. In the folder *other* are stored all other tries that were not successful and therefore were not included in the document. The file NLP-EDOS-SVM.ipynb contains the notebook of the implementation described in section 3.2.1. The file NLP-EDOS-BERT.ipynb contains the notebook of the implementation described in section 3.2.2. The file NLP-EDOS-ELECTRA.ipynb contains the notebook of the implementation described in section 3.2.3.

4. Evaluation

The evaluation is performed through the use of metrics:

- **Accuracy:** Overall effectiveness of the model, calculated as $\frac{TP+TN}{TP+TN+FP+FN}$.
- **Precision:** The ratio of true positive predictions to the total predicted positives, calculated as $\frac{TP}{TP+FP}$. Precision measures how many of the predicted positive cases were actually positive.
- **Recall:** The ratio of true positive predictions to the total actual positives, calculated as $\frac{TP}{TP+FN}$. Recall measures how many of the actual positive cases were correctly predicted by the model.
- **F1 Score:** The harmonic mean of precision and recall, calculated as $2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$. The F1 score provides a balance between precision and recall.

Furthermore, we make use of confusion matrices for greater clarity and detail of the classification carried out, in order to better analyze the models. The confusion matrix can be visualized using libraries like Matplotlib and Scikit-learn in Python.

Also, time of execution should be considered as an important metric, however, it is strongly effected by our run environment. See section 3.2 for more details.

4.1. Baseline SVM

Evaluation on the Test set:

- **Accuracy:** 0.600
- **Precision:** 0.804
- **Recall:** 0.600
- **F1 Score:** 0.573
- **Time of execution:** around 15 seconds for the execution of the SVM model.

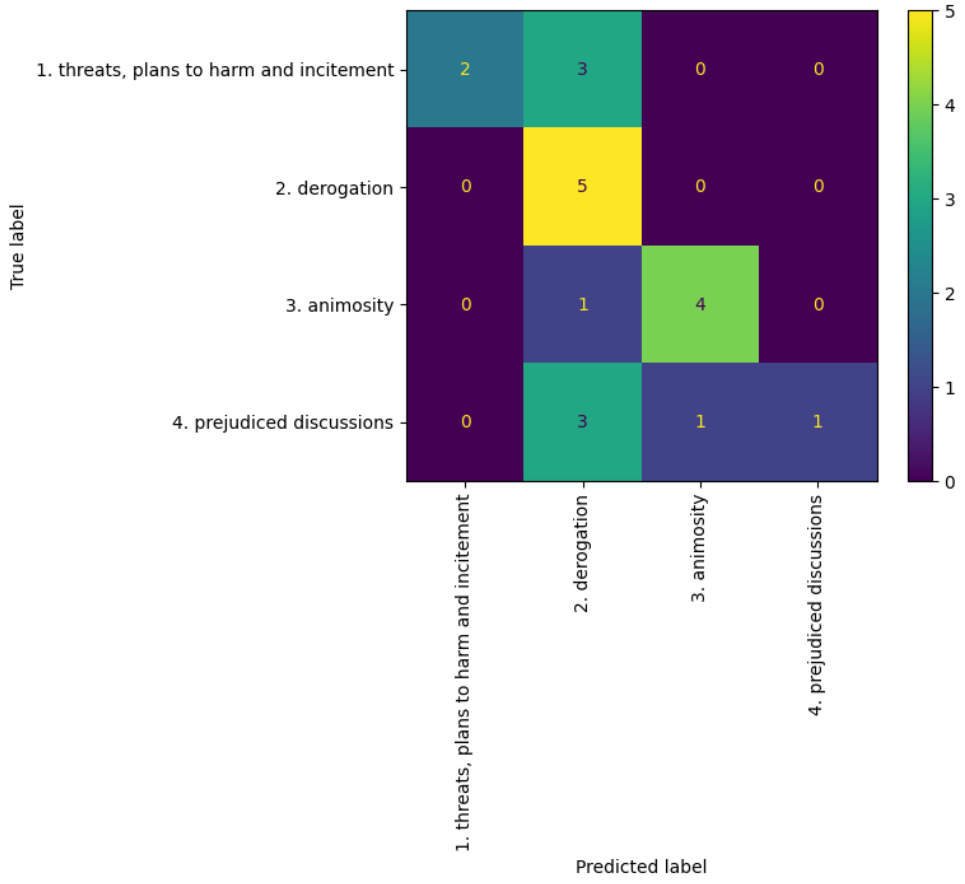

Figure 1: Confusion Matrix of the SVM model on the Test set

See Fig.1 for the confusion matrix of the SVM model run: as we can see, the unbalanced dataset confuses a lot the classifier: it tends to classify more on category 2. The metric of F1 score is around 0.5 and the recall is low, this suggests that the model struggles either with the sensitivity measure (low true positive rate), indicating a higher number of misclassifications. The model has a good precision.

4.2. BERT

Evaluation on the Test set:

- **Accuracy:** 0.650
- **Precision:** 0.718
- **Recall:** 0.650
- **F1 Score:** 0.645
- **Time of execution:** around 1 hour and 10 minutes per epoch.
- **Time of execution (total):** around 3 hour and 30 minutes for the entire training (1 epoch per fold, 3 fold).

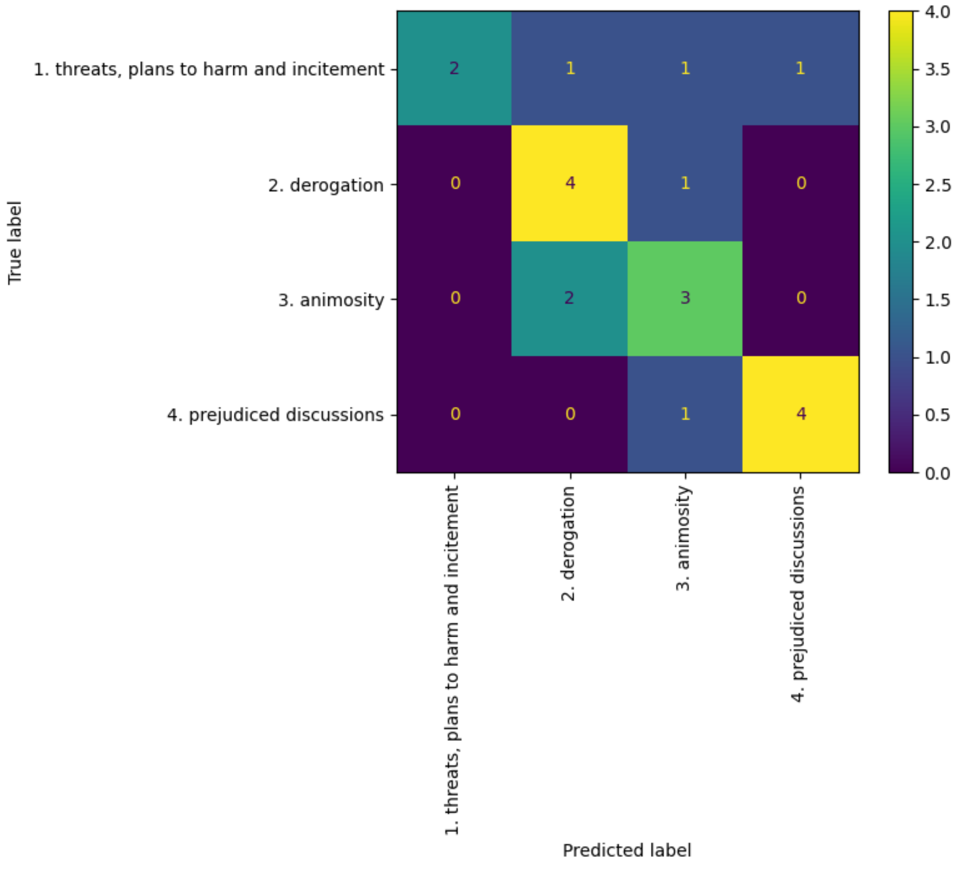

Figure 2: Confusion Matrix of the BERT model on the Test set

See Fig.2 for the confusion matrix of the BERT model run: as we can see, the problem of the unbalanced dataset is solved with this model but struggles a bit with misclassifications, having a f1 score around 0.6, indicating a little improvement wrt SVM baseline, however it has a lower precision.

4.3. Electra

Evaluation on the Test set:

- **Accuracy:** 0.800
- **Precision:** 0.816
- **Recall:** 0.800
- **F1 Score:** 0.804
- **Time of execution:** around 14 minutes per epoch.
- **Time of execution (total):** around 3 hour and 30 minutes for the entire training (5 epoch per fold, 3 fold).

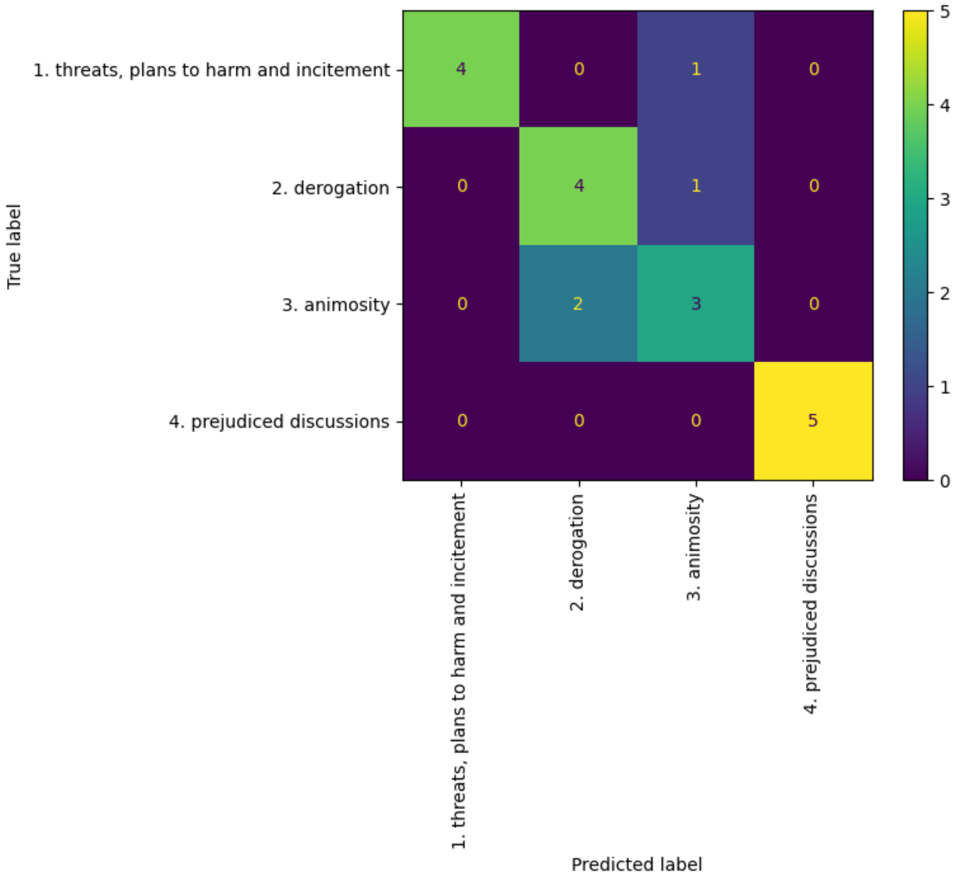

Figure 3: Confusion Matrix of the ELECTRA model on the Test set

See Fig.3 for the confusion matrix of the ELECTRA model run: as we can see, the confusion matrix appears to be highly diagonalized, indicating an good performance at the same time compared to the BERT model run (4.2), confirmed by the higher f1 score.

5. Conclusions and Limitations

In this project, we evaluated the performance of text categorization models, including the Electra model, BERT model, and a Support Vector Machine (SVM) used as a baseline. Our experiments demonstrated that the Electra model outperformed both the BERT model and the SVM classifier across multiple evaluation metrics, including F1 score, precision, and recall. The superior performance of Electra can be attributed to its efficiency in capturing contextual information [2].

With the same execution time (approximately), we can see how much more effective the ELECTRA model is compared to the basic BERT model. The long duration of the BERT model execution might be due to issues related to our personal computer environment or the virtual machines used on Google Collab, as the performance of the latter is not stable on the free plan.

5.1. Limitation of the preprocessing

We've also done some run with an advanced preprocessing with ngrams and word embedding (not mentioned here), but those techniques did not bring hoped-for improvements, on the contrary they made the classifier null (See section 3.3)

An explaination could be that with the Ngram analysis the feature space becomes extremely sparse, making it difficult for the model to learn meaningful patterns, especially if the dataset is small. Also, many bigrams or trigrams might not appear frequently enough to contribute significantly to the classification decision, leading to overfitting on the training data and poor generalization on the test data. The lack of fine tuning on the model and the unbalanced dataset could be the causes of the failure of both techniques.

References

[1] Hannah Rose Kirk, Wenjie Yin, Bertie Vidgen, and Paul Röttger. SemEval-2023 Task 10: Explainable Detection of Online Sexism. 2023. *arXiv:2303.04222 [cs.CL]*. https://arxiv.org/abs/2303.04222.

[2] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. 2020. *arXiv:2003.10555 [cs.CL]*, https://arxiv.org/abs/2003.10555.

[3] Hugging Face. ELECTRA Model Documentation. https://huggingface.co/docs/transformers/en/model_doc/electra, 2023.

[4] Claudio Mazzoni. Towards Data Science. Augment your small dataset using transformers: Synonym replacement for sentiment analysis (part 1). https://towardsdatascience.com/augment-your-small-dataset-using-transformers-synonym-replacement-for-sentiment-analysis-part-1-87a838cd0baa, 2020.

[5] Scikit-learn. Support Vector Machines. https://scikit-learn.org/stable/modules/svm.html, 2023.

[6] Scikit-learn. Cross-validation: evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html, 2020.

[7] Thilina Rajapakse. Towards Data Science. Battle of the Transformers: ELECTRA, BERT, RoBERTa, or XLNet. https://towardsdatascience.com/battle-of-the-transformers-electra-bert-roberta-or-xlnet-40607e97aba3, 2020.

[8] Simone Innocenti. Thesis: Sentiment analysis in context: Investigating the use of BERT and other techniques for ChatBot improvement. University of Padua. Supervisor: Giovanni Da San Martino. https://hdl.handle.net/20.500.12608/50232, 2023.

[9] Hugging Face. Sequence Classification with Transformers. https://huggingface.co/docs/transformers/tasks/sequence_classification, 2023.

[10] Furkan Ayik. Mastering Text Classification with BERT: A Comprehensive Guide. *Medium*, 2023. https://medium.com/@ayikfurkan1/mastering-text-classification-with-bert-a-comprehensive-guide-194ddb2aa2e5.