**HIT**

**M.Sc. – Data science**

**Toxic comments classification project – Final Report**

**NLP Course: 99101**

**Written By:**

Idan Vazana, 204154207

Roni Shternberg, 034466862

June, 2023



## **TABLE OF CONTENTS**

[TABLE OF CONTENTS 2](#_Toc138501393)

[CHART OF ILLUSTRATIONS 2](#_Toc138501394)

[1. EXECUTIVE SUMMARY 3](#_Toc138501397)

[2. CHALLENGES & MOTIVATION 4](#_Toc138501398)

[3. REVIEW OF LITERATURE 6](#_Toc138501399)

[4. RESEARCH QUESTION & CHALLENGES 8](#_Toc138501400)

[5. WORK PLAN & METRICS 9](#_Toc138501401)

[6. EDA 10](#_Toc138501402)

[7. MODELS & SCORE 15](#_Toc138501403)

[8. MODELS EVALUATE 23](#_Toc138501404)

[9. CONCLUSION 25](#_Toc138501405)

[10. BIBILIOGRAPHY 26](#_Toc138501406)

## **CHART OF ILLUSTRATIONS**

[Figure 1 – TOXIC / NON-TOXIC COMMENTS 6](file:////Users/idanvazana/Library/Mobile%20Documents/com~apple~CloudDocs/M.sc%20-%20Data%20Science/סמסטר%20ב/NLP/פרויקט/Toxic%20Comments%20Classification%20Project.docx#_Toc138501605)

[Figure 2- TOXIC COMMENTS DISTIBUTION 9](#_Toc138501606)

[Figure 3 – WORK PROCESS 10](file:////Users/idanvazana/Library/Mobile%20Documents/com~apple~CloudDocs/M.sc%20-%20Data%20Science/סמסטר%20ב/NLP/פרויקט/Toxic%20Comments%20Classification%20Project.docx#_Toc138501607)

[Figure 4 – TOXIC CLASS COUNT 11](file:////Users/idanvazana/Library/Mobile%20Documents/com~apple~CloudDocs/M.sc%20-%20Data%20Science/סמסטר%20ב/NLP/פרויקט/Toxic%20Comments%20Classification%20Project.docx#_Toc138501608)

[Figure 5 – DATA PIE CHART 11](file:////Users/idanvazana/Library/Mobile%20Documents/com~apple~CloudDocs/M.sc%20-%20Data%20Science/סמסטר%20ב/NLP/פרויקט/Toxic%20Comments%20Classification%20Project.docx#_Toc138501609)

[Figure 6 – CORRELATION MATRIX HEATMAP 12](#_Toc138501610)

[Figure 7 – TEXT TO DOC 13](#_Toc138501611)

[Figure 8 - TOKENIZATION EXAMPLE 13](#_Toc138501612)

[Figure 9 – DATA COMPARATION PLOT (PRE AND POST PROCESS) 14](file:////Users/idanvazana/Library/Mobile%20Documents/com~apple~CloudDocs/M.sc%20-%20Data%20Science/סמסטר%20ב/NLP/פרויקט/Toxic%20Comments%20Classification%20Project.docx#_Toc138501613)

[Figure 10 – WORD CLOUD PLOT 15](#_Toc138501614)

[Figure 11 – NUMBER OF WORDS IN A SENTENCE 20](file:////Users/idanvazana/Library/Mobile%20Documents/com~apple~CloudDocs/M.sc%20-%20Data%20Science/סמסטר%20ב/NLP/פרויקט/Toxic%20Comments%20Classification%20Project.docx#_Toc138501615)

[Figure 12 – BERT LAYERS 21](file:////Users/idanvazana/Library/Mobile%20Documents/com~apple~CloudDocs/M.sc%20-%20Data%20Science/סמסטר%20ב/NLP/פרויקט/Toxic%20Comments%20Classification%20Project.docx#_Toc138501616)

[Figure 13 – CLASSIFICATION REPORT 24](#_Toc138501617)

[Figure 14 - CONFUSSION MATRIX 24](#_Toc138501618)

רונית שמלו

## EXECUTIVE SUMMARY

In our project in the Natural Language Processing course, we chose to focus on and investigate the challenge of text classification. Our goal is to identify and classify online toxic comments from the Kaggle “Jigsaw toxic comment classification challenge” competition (i.e., comments that are rude, disrespectful, or otherwise likely to make someone leave a discussion).

The data set contains 160K Wikipedia talk page comments. The data is imbalanced, and we deal with it during the project.

Our research discusses the multi-label classification of toxic and anti-social behavior written by users on the Wikipedia platform into serval types of toxicity in the English language: Toxic, Severe toxic, Obscene, Threat, Insult, and Identity hate.

At first, by the recommendation of the lecture we classified the data into two classes and keep the others to further challenge.

Wikipedia is a multilingual project to compile a collaborative, free, and reliable encyclopedia that everyone can edit. Today there are 6,636,852 articles in the English language and every day 555 new articles are created.

The talk page includes discussions about the content of the page it is attached to. The purpose of the talk page is to write comments from the users on wrong, defective, controversial, incorrectly worded text, and more.

The discussion on the talk pages is important to improve the quality of the article and discuss fundamental questions in its context.

Also, if a user deviates from the rules of conduct, a warning note will be created against him and next time even a block.

After conducting an extensive literature review, which includes scientific articles, study materials, and articles (references in the last chapter). We learned that with the help of a Transformer-based model, researchers achieved the highest benchmarks. In our project we applied an EDA process for data research, and data visualization and trained 3 classified models: BERT, MultinomialNB, and Linear SVC. In the project, you can get an impression of comparing indices for models and our recommendations for further research.

## **CHALLENGES & MOTIVATION**

While researching and exploring the topic of toxic comments text classification we learned about stages of natural language processing projects, methods, and algorithms.

The threat and insults people experience because of their comments on social media and other platforms, in general, are ever-increasing. This causes many people to think twice or even stop expressing themselves on digital platforms to avoid having to deal with different opinions.

Some mechanisms are activated by the platform managers to identify toxic trends to facilitate and improve the nature of the conversation on the network.

An example of this is the reporting of offensive content by different users.

The downside of this method is that there is still some level of exposure to the written content.

In the era of protests and freedom of expression, there is a great potential for damage to the networks.

There are many articles and publications of incidents on the topic of loss, threats, and boycott of teenagers, in some of the findings it is evident that a trend of toxic reactions on the networks could be identified around them.

Thus, the main challenge in our project is the classification of the toxic content into several types of toxicity.

Another challenge is ambiguity - a comment may be classified as offensive because a word is recognized but there is a mistake in its context.

We can see examples of toxic comments in the following figure:

Table

Description automatically generated

Figure 1 – TOXIC / NON-TOXIC COMMENTS

## **REVIEW OF LITERATURE**

The first article compares the effectiveness of various deep learning models and text preprocessing techniques for classifying toxic comments. The study uses a dataset of comments from Wikipedia that have been labeled as toxic or non-toxic. The text preprocessing techniques include stopword removal, stemming, and lemmatization. The deep learning models tested include convolutional neural networks (CNNs), recurrent neural networks (RNNs), and a hybrid model that combines both CNNs and RNNs.

The results of the study show that the hybrid CNN-RNN model outperforms the other models and achieves the highest accuracy in classifying toxic comments. Among the text preprocessing techniques, lemmatization improves the performance of all models, while stopword removal and stemming have mixed effects depending on the model used.

Overall, the study suggests that a combination of deep learning models and appropriate text preprocessing techniques can effectively classify toxic comments.

***Viera Maslej, Martin Sarnovsky, Peter Butka and Kristina Machova, University of Kosice, 2020, Comparison of deep learning models and Various text pre-processing techniques for the toxic comment’s classification.***

The second article "Text Classification Algorithms: A Survey" provides a comprehensive overview of various text classification algorithms. The authors begin by introducing the concept of text classification and its applications, such as document categorization, sentiment analysis, and spam filtering.

The article then discusses various text representation methods, including bag-of-words, n-grams, and word embeddings. The authors explain the advantages and disadvantages of each method and their suitability for different text classification tasks.

Next, the article reviews several traditional machine learning algorithms commonly used in text classification, such as Naive Bayes, Support Vector Machines, and Decision Trees. The authors discuss the strengths and weaknesses of each algorithm and their application in text classification tasks.

The article also examines deep learning approaches for text classification, such as Convolutional Neural Networks, Recurrent Neural Networks, and Attention Mechanisms. The authors explain the advantages of these models and their performance compared to traditional machine learning algorithms.

Finally, the article concludes with a discussion of future directions in text classification research, including the development of hybrid models that combine traditional machine learning algorithms with deep learning approaches.

Overall, "Text Classification Algorithms: A Survey" provides a thorough and insightful overview of the various text classification algorithms and their applications.

***Kamram Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, Donald Brown, Multidisciplinary Digital Publishing Institute, 2019, Text Classification Algorithms: A Survey.***

The third article describes a complete process for building a text classification system using state-of-the-art natural language processing (NLP) models. The process includes several stages, such as data collection and preprocessing, feature extraction, model selection, and evaluation.

The authors apply several NLP models, including Word2Vec, GloVe, FastText, and BERT, to classify text data from the 20 Newsgroups dataset and the Amazon Customer Reviews dataset. They compare the performance of the models using several evaluation metrics, including accuracy, precision, recall, and F1-score.

The results show that the BERT model outperforms the other models in both datasets, achieving an accuracy of over 90%. The authors also provide insights into the strengths and weaknesses of each model and discuss the potential applications of text classification in various fields, such as social media analysis and customer feedback analysis.

***Varun Dogra, Sahil Verma, Kavita, Pushpita Chatterjee, Jana Shafi, Jaeyoung Choi, Muhammad Fazal Ijaz, Computational Intelligence and Neuroscience, 2022, A Complete Process of Text Classification System Using State of the Art NLP Models.***

## **RESEARCH QUESTION & CHALLENGES**

Does a comment written by a user on a Wikipedia entry's talk page content is targeted as toxic content that is avoided with one of the following types:

• Toxic – Very bad, unpleasant, harmful.

• Severe toxic – Extremely bad and offensive.

• Obscene – Description of sexual matters.

Toxic

• Threat – a statement of an intention to inflict pain, injury, or damage.

• Insult – speak to or treat with disrespect or scornful abuse.

• Identity hate – hatred, hostility, or violence towards ethnicity, nation, religion, gender

Unfortunately for the problem, but fortunately for the Wikipedia community, toxic comments are rare. Just over 10% of this data set is labeled as toxic, but some of the subcategories are extremely rare and make up less than 1% of the data.

Because of this imbalance, accuracy is an almost useless metric for evaluating classifiers for this problem.

In our project, we united the 5 departments into one called Non-Toxic.

Additionally, we suggest using F1 Score, which severely penalizes models that simply predict everything as positive or negative with an unbalanced data set.

The F1 score is a harmonic mean between precision and recall. It combines the strengths of precision and recalls while balancing their weaknesses, creating a score that can fairly estimate models regardless of dataset imbalances.

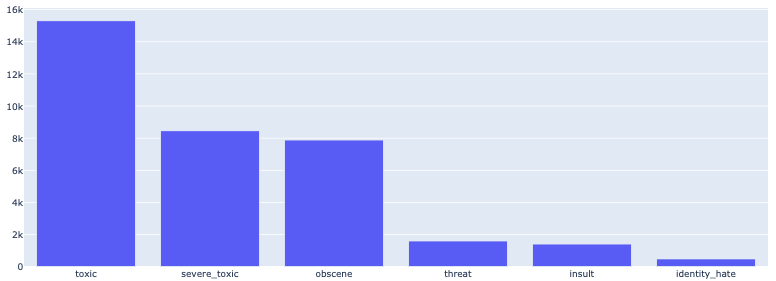


Figure 2- TOXIC COMMENTS DISTIBUTION

## **WORK PLAN & METRICS**

In our project, we applied natural language processing modules and algorithms to classify the comments into different categories.

The following diagram describes the phases of work in the project.

After choosing the challenge, we turned to the professional literature to review articles on various topics such as:

Understanding the data

Data cleaning and uniformity

The presentation of the data

Comparison between text classification models

Comparison between programming practices and more.

The project report is described according to the stages of our research work.

A picture containing electronics

Description automatically generated

Figure 3 – WORK PROCESS

Programming and Metrics:

Google collab.

Python – NumPy, pandas, matplotlib, Spacy, sklearn, pytorch.

GPU – A100

Models – BERT, MultinomialNB, LinearSVC, Logistic Regression.

## **EDA**

Text cleaning and preprocessing

In general, in natural language processing (NLP), most text and documents contain many unnecessary words for text classification: Stop words, misspellings, slang, etc. In this part, we learned about patterns and pre-processed the data.

The data preparation phase is a crucial element for extracting the features that will form the input layer in the text classification models. When this process is carried out accurately, the evaluation measures of the models are even more reliable.

We have made use of different algorithms to perform these techniques.

In this report we will include the most prominent charts from the techniques we applied, you can get an impression of each process and the charts detailed in depth in the code notebook.

* 1. We added two columns that describe the toxic level or non-toxic comments – “toxic class count” & “is toxic”.

Figure 4 – TOXIC CLASS COUNT

|  |  |
| --- | --- |
| **Test** | **Result** |
| Nulls | Non-nulls |
| Duplicates | Non-duplicates |

A blue and red pie chart

Description automatically generated with medium confidence

* 1. Pie chart – Toxic / Non-Toxic comments:

Toxic – 16,225

Non-Toxic – 143,346

Figure 5 – DATA PIE CHART

* 1. Correlations Matrix Heatmap:

We can see a correlation of 0.74 between obscene and insulting comments.

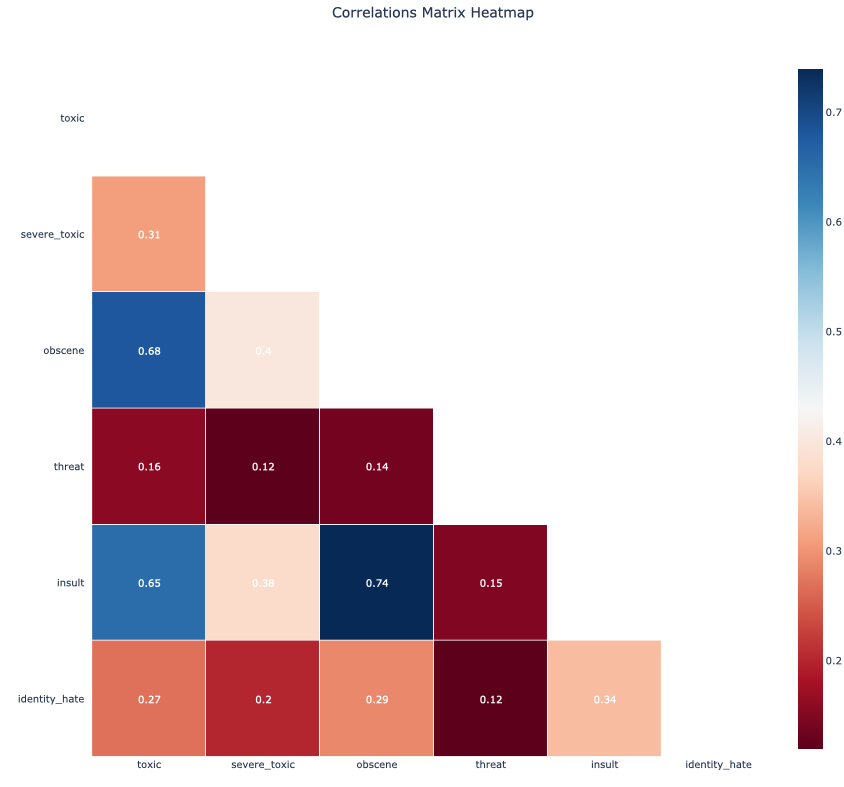


Figure 6 – CORRELATION MATRIX HEATMAP

* 1. Tokenization:

Tokenization is the process of breaking down a stream of text into words, phrases, symbols, or any other meaningful element. The main purpose of this step is to extract individual words in the sentence and convert them to a variable of type doc.

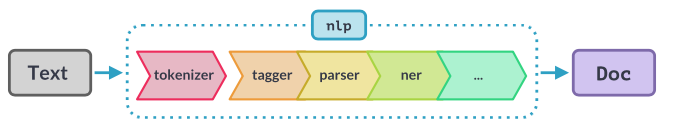


Figure 7 – TEXT TO DOC

* INTJ – A Word or phrase used to express strong emotion or surprise.
* Noun: A word that represents a person, place, thing, or idea.
* Pronoun (PRON): A word used in place of a noun to refer to a person or thing.
* Adverb (ADV): A word that modifies a verb, adjective, or other adverb.
* Verb: A word that expresses an action, occurrence, or state of being.
* Punctuation (PUNCT): Is used to identify punctuation marks in a sentence.
* Auxiliary Verb (AUX): The "AUX" tag is used to represent auxiliary verbs. Auxiliary verbs are verbs that accompany the main verb in a sentence and help to express grammatical aspects such as tense, mood, voice, or aspect. Examples of auxiliary verbs include "be," "have," "do," "will," "can," etc.

For example, in our project:

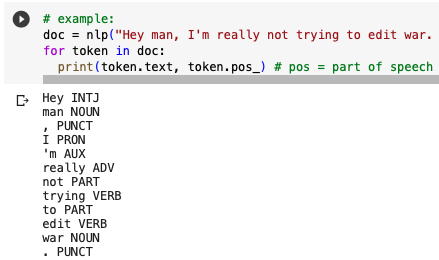


Figure 8 - TOKENIZATION EXAMPLE

* 1. Lemmatization:

Text lemmatization is the process of eliminating redundant prefixes or suffixes of a word and extracting the base word (lemma).

These attributes and methods are used in the clean function to filter out certain types of tokens from the data.

* 1. Stop words:

Stop words are typically removed from text data during preprocessing to improve computational efficiency and focus on more meaningful words stop words refer to commonly used words that are considered insignificant or do not carry significant meaning or context. Examples of stop words in English include "the", "is", "in", "and", "at", "it", and "on".

* 1. Token like URL:

This attribute checks if the token resembles a URL or web address.

* 1. Token is digit:

This attribute checks if the token consists of only digits.

* 1. Token is currency:

This attribute checks if the token represents a currency symbol or a currency-related word.

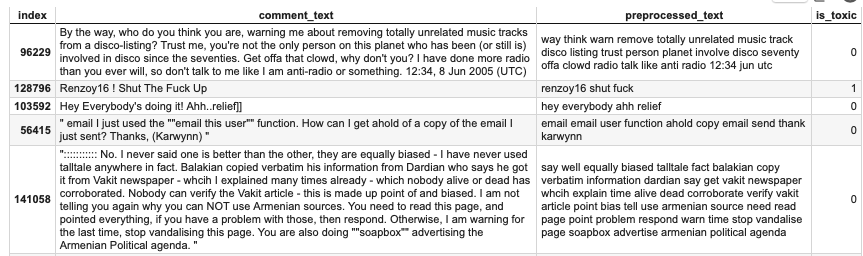


Figure 9 – DATA COMPARATION PLOT (PRE AND POST PROCESS)



Figure 10 – WORD CLOUD PLOT

## **MODELS & SCORE**

From a perusal of the articles noted in the bibliography, and in the source code and the results are shared as free tools https://github.com/kk7nc/Text\_Classification It is evident that a combination of deep learning models and machine learning models gives the maximum accuracy.

As part of the process of preparing the data for the model and feature extractor.

There are a few metrics like Word Embeddings, CountVectorizer, and more. We chose to use the TFidfVectorizer model to extract the words into a matrix.

The TFidfVectorizer converts a collection of raw text documents into a matrix of numerical features. It takes care of two key aspects:

Term Frequency (TF): It calculates the frequency of each term (word) within a document. The assumption is that the more frequently a term appears in a document, the more important it is to that document.

Inverse Document Frequency (IDF): It measures the rarity or uniqueness of a term across the entire document corpus. The assumption is that terms that occur in fewer documents are more informative or significant.

We defined a function named 'print result' that is activated at the end of each model and calculates the accuracy, precision, recall, and F1 score, Almost the function prints the classification report and confusion matrix.

* 1. Score Explanation
     1. Accuracy measures the overall correctness of the model's predictions by calculating the ratio of correctly classified samples to the total number of samples.

It provides an overall assessment of the model's performance.

* + 1. Precision measures the proportion of correctly predicted positive instances (true positives) out of all instances predicted as positive (true positives + false positives). Precision focuses on the accuracy of the positive predictions and is a measure of how well the model avoids false positives.
    2. Recall, also known as sensitivity or true positive rate, measures the proportion of correctly predicted positive instances (true positives) out of all actual positive instances (true positives + false negatives). Recall focuses on the ability of the model to identify all positive instances and is a measure of how well the model avoids false negatives.
    3. The F1 score is the harmonic mean of precision and recall. It provides a balanced measure that considers both precision and recall. The F1 score is useful when you want to find a balance between precision and recall, as it considers both false positives and false negatives. It ranges from 0 to 1, with 1 being the best possible score.
  1. Model Explanation
     1. Logistic regression can handle large scale text classification tasks with large number of features (words in comments).

However, the performance of logistic regression for text classification depends on the quality of the features (word representations), size and quality of the training data. Also, there is a very high sensitivity to abnormal data and apparently the results of models built on the principle of neural networks or transformers will have better performance.

Logistic regression is computationally efficient and can handle large-scale text classification tasks with large numbers of features (words) and documents.

Also, it provides interpretable results by estimating coefficients for each feature (word) that indicate the effect of that feature on the classification decision. These coefficients can provide insight into the important words associated with toxic or non-toxic comments. Despite the model is a linear model, logistic regression can capture non-linear relationships between traits and the target variable by using appropriate trait transformations, interactions, or combining higher-order terms.

Logistic regression has a relatively low complexity compared to more complex algorithms such as neural networks (in our project the BERT model), which makes it easier to implement, train and interpret.

After performing data preparation and TF-IDF vectorizer, we used the training data for a logistic regression model. It can be seen from the parameters for evaluating the model that its performance was indeed among the best in the project and moved to the LinearSVC model. Which also indicates a good preparation for the data.

* + 1. Linear Support Vector Classifier is a popular classification model used for text classification tasks, including binary classification problems and therefore also suitable for our topic of classifying text as toxic or non-toxic.

Linear SVC aims to find the best plane that separates the two classes (toxic and non-toxic) by maximizing the space between the classes. The margin represents the distance between a hyperplane and the nearest data points from each class.

Linear SVC assumes that the classes can be separated by a linear decision boundary in the feature space. This works well when there is a clear linear separation between toxic and non-toxic texts.

Efficient training: Linear SVC is computationally efficient, especially compared to non-linear SVC models. It scales well to large datasets with many features, making it suitable for text classification tasks involving large numbers of documents and words.

The model in our project has achieved very good performance and ranks second in number among the other models used.

* + 1. Multinomial Naive Bayes is a popular algorithm used for text classification tasks. This is a variant of the naive Bayes algorithm, which assumes that the features are generated from a multinomial distribution.

In our context of toxic/non-toxic comment classification, MultinomialNB works by building a probabilistic model based on the frequency counts of words in the training data. It calculates the likelihood of each word appearing in toxic and non-toxic comments, as well as the prior probabilities of a comment being toxic or non-toxic.

Using Bayes theorem, it calculates the posterior probabilities of a note being toxic or non-toxic given its word frequencies.

During classification, MultinomialNB assigns an annotation to the class (toxic or non-toxic) with the highest posterior probability. It assumes independence between the occurrences of different words in a note, which is a simplifying assumption known as the "naive" part of naive Bayes.

Despite its simplifying assumptions, MultinomialNB often performs well in practice, especially for text classification tasks with sparse data. It is computationally efficient, scales well to large datasets, and can handle high-dimensional feature space. However, it may have difficulty capturing complex relationships between words and may not be suitable for tasks where word order or context is important as in this case:

"The cat chased the dog."

"The dog chased the cat."

Overall, MultinomialNB is popular for these tasks, in this project its performance was the lowest, you can get an impression of the model performance evaluation in the next chapter.

* + 1. BERT – By stacking multiple transformer layers, BERT can effectively model long-range dependencies, capture intricate language patterns, and generate rich contextualized representations. This depth allows BERT to achieve state-of-the-art performance on various natural language processing tasks and enables transfer learning across different domains.

Unlike the previous models, for this model, it is required to use a specific tokenizer suitable for the model.

Before we are ready to encode our text, we need to decide on a maximum sentence length for padding/truncation.

BERT has two constraints:

* All sentences must be padded or truncated to a single, fixed length.
* The maximum sentence length is 512 tokens.

That's why we measured the length of the cells in general and the maximum cell specifically.

Max sentence length (tokens): 1987

Total number of tokens in all sentences: 618659

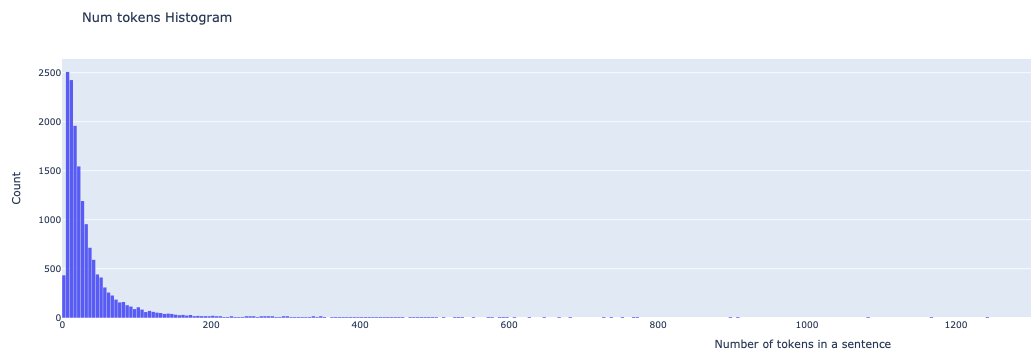


Figure 11 – NUMBER OF WORDS IN A SENTENCE

In the tokenizer phase, the BERT model creates pads for the purpose of completing the matrix for sentences of uniform length, where the label 1 indicates that there is a word and the label 0 indicates that there is no word in the sentence, therefore these labels will not be considered for processing the text.

* + - 1. How does it work?
* Self-Attention Mechanism: The self-attention mechanism allows each word/token in the input sequence to attend to other words/tokens in the same sequence. It helps capture dependencies and relationships between different words, enabling BERT to understand the contextual information effectively.
* Encoding Contextualized Representations: Each transformer layer takes in the input tokens and generates contextualized representations for each token. These representations incorporate information from both preceding and following tokens, allowing BERT to capture the context in which each word appears.
* Feature Extraction: Each transformer layer in BERT extracts higher-level features from the input tokens. The lower layers focus on low-level linguistic features, such as word forms and syntax, while the higher layers capture more abstract and semantic information.
* Fine-tuning and Transfer Learning: The multiple layers in BERT provide a hierarchical representation of the input text. After pretraining on large-scale corpora, the pre-trained BERT model can be fine-tuned on specific downstream tasks, such as sentiment analysis or named entity recognition. The layers closer to the input can capture task-independent features, while the deeper layers can adapt to task-specific patterns during fine-tuning.

Each transformer layer involves multiple sub-layers, such as self-attention and feed-forward neural networks, which contribute to the overall transformation of the input tokens at each layer.

A screenshot of a computer

Description automatically generated with low confidence

Figure 12 – BERT LAYERS

* + - 1. Highlights and comments from the code:

The model.train() function is typically used in PyTorch to set the model in training mode.

When you call model.train(), it activates certain modules in the model that behave differently during training compared to evaluation/testing.

These modules include dropout and batch normalization layers.

During training, dropout layers randomly set a fraction of their inputs to zero,

which helps in regularization and prevents overfitting.

Batch normalization layers calculate batch-wise statistics and normalize the inputs, aiding in faster convergence and improved generalization.

By default, PyTorch sets the model to training mode when you create an instance of it.

However, it's good practice to call model.train() explicitly before the training loop to ensure that the model is indeed in the training mode.

After the training loop, you can call model.eval() to switch the model to evaluation mode.

In the evaluation mode, the behavior of dropout and batch normalization layers changes.

Dropout layers no longer drop inputs, and batch normalization layers use the running statistics computed during training instead of calculating batch-wise statistics.

It's essential to set the model to the appropriate mode (train() or eval()),

as it ensures that the correct behavior is applied to the model's modules during training and evaluation/testing, respectively.

As we expected, we can see very high results from training the model, of course, if we had chosen to run more epochs, we might have reached even higher results.

## **MODELS EVALUATE**

The logistic regression model achieves a high accuracy of 0.93, indicating that it correctly classifies 93% of the time in your toxic/non-toxic classification task. It also has a high precision of 0.96, which implies that when it predicts a comment as toxic, it is correct 96% of the time. However, the recall and F1 scores are relatively low, 0.37 and 0.53, respectively, indicating that the model has difficulty identifying all toxic comments correctly.

The linear SVC model performs slightly better than logistic regression with an accuracy of 0.94. It shows a good balance between a precision of 0.84 and a recall of 0.59, resulting in an F1 score of 0.69. This suggests that the model maintains a reasonable trade-off between correctly identifying toxic annotations and minimizing false positives.

MultinomialNB achieves an accuracy of 0.92, indicating an overall high classification performance for your toxic/non-toxic classification task. It exhibits perfect accuracy 1.0, meaning it does not produce false positive predictions. However, the recall is relatively low at 0.2, indicating that the model has difficulty identifying a significant proportion of the toxic comments. As a result, the F1 score is 0.39, reflecting the unbalanced performance between precision and recall.

BERT outperforms the other models with a high accuracy of 0.96 in our project. It exhibits good precision of 0.8 and recall of 0.83 values, resulting in an F1 score of 0.81. This indicates that BERT achieves a strong balance between precision and recall, making it well-suited for the accurate classification of toxic and non-toxic annotations.

In summary, BERT stands out as the best performer, with high precision, a solid balance between precision and recall, and an excellent F1 score. Furthermore, it is necessary to consider other factors such as computational resources, data size, and time which the BERT model also required a lot of adjustment.

Figure 13 – CLASSIFICATION REPORT

On the Confusion matrix logistic regression and MultinomialNB show high true positive rates but struggle with false negatives, while linear SVC achieves a reasonable balance.

BERT turns out to be a strong performer with a good balance of true positives, true negatives, false positives, and false negatives.

Based on these observations, we can conclude that BERT is the most promising model for the accurate classification of toxic and non-toxic comments.

Figure 14 - CONFUSSION MATRIX

## **CONCLUSION**

Throughout the semester, during the lessons and from the work on the project, we had the opportunity to research and discuss quite a few articles, articles, and exercises that deal with different practices for text classification.

It is evident that this field is gaining momentum and meets many fields in the industry such as law, economics, medicine, and more.

The learning within the project enriched her with practically applied tools to implement the text analysis and classification process.

Starting with the stages of learning the data and comparing them to parallel projects, the process of preparing the data is very different from working with numerical data and ending with the realization of the models with an emphasis on the BERT model and the adjustment of the parameters to extract effective results.

The joint work contributed to mutual learning and maximum exploitation of the project as a teaching act.

There are several topics that we would recommend for further research:

* 1. Trial and error small & large SpaCy.
  2. Classification of the toxic comments department into 6 sub-departments as proposed at the beginning of the project.
  3. Testing whether correcting spelling errors improves the result.
  4. An attempt to filter the word Wikipedia from the sentences (in a word cloud you can see that it is very significant).
  5. Testing whether removing the word Wikipedia from the sentences will affect the results (in a word cloud you can see that this is very significant).
  6. TextCategorizer – Use the data in this model.
  7. Hyper-Parameter Tunning (eg Grid Search).
  8. Cross-Validation.
  9. BERT Cased/Uncased

We hope for the day when there will be an accessible and available feature to filter offensive comments on social networks for the end users, the social space will perhaps become a little more sensitive.

We would like to thank the course staff for their support throughout the project.

## **BIBILIOGRAPHY**

**Article 1:**

Viera Maslej, Martin Sarnovsky, Peter Butka and Kristina Machova, University of Kosice, 2020, Comparison of deep learning models and Various text pre-processing techniques for the toxic comment’s classification.

<https://www.mdpi.com/2076-3417/10/23/8631>

**Article 2:**

Kamram Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, Donald Brown, Multidisciplinary Digital Publishing Institute, 2019, Text Classification Algorithms: A Survey.

<https://www.mdpi.com/2078-2489/10/4/150>

**Article 3:**

Varun Dogra, Sahil Verma, Kavita, Pushpita Chatterjee, Jana Shafi, Jaeyoung Choi, Muhammad Fazal Ijaz, Computational Intelligence and Neuroscience, 2022, A Complete Process of Text Classification System Using State of the Art NLP Models.

<https://www.hindawi.com/journals/cin/2022/1883698/>

**Talk Wikipedia page explanation:**

<https://en.wikipedia.org/wiki/Help:Talk_pages>

**Data set of toxic comments from Kaggle:**

<https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/overview>

**Lemmatization vs Stemming:**

<https://www.analyticsvidhya.com/blog/2022/06/stemming-vs-lemmatization-in-nlp-must-know-differences/>

<https://stackoverflow.com/questions/1787110/what-is-the-difference-between-lemmatization-vs-stemming>

**NLTK vs spaCy:**

<https://www.kaggle.com/general/299223>

<https://www.activestate.com/blog/natural-language-processing-nltk-vs-spacy/>

<https://proxet.com/blog/spacy-vs-nltk-natural-language-processing-nlp-python-libraries/>

**Introduction to Word Embedding and Word2Vec:**

<https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa>

**Natural Language Processing Tutorial - From Data Camp:**

<https://www.datacamp.com/tutorial/tutorial-natural-language-processing>

<https://www.datacamp.com/blog/how-nlp-is-changing-the-future-of-data-science>

**Bag of Words article:**

<https://machinelearningmastery.com/gentle-introduction-bag-words-model/>

**TF-IDF:**

<https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

**Word2Vec vs GloVe – A Comparative Guide to Word Embedding Techniques:**

<https://analyticsindiamag.com/word2vec-vs-glove-a-comparative-guide-to-word-embedding-techniques/>

**BERT model:**

**https://curiousily.com/posts/sentiment-analysis-with-bert-and-hugging-face-using-pytorch-and-python/**

https://mccormickml.com/2019/07/22/BERT-fine-tuning/