Project Report

Kemal Şahin

Student Number: 2200765021

Burak Kurt

Student Number: 2200765010

Course Name: AIN420

Contents

1 Introduction

1.1 Project Overview

In the digital age, social media platforms and online forums have become central to public discourse, providing a space for individuals to express their thoughts, opinions, and emotions. While these platforms enable free expression and the exchange of ideas, they also pose significant challenges in content moderation, particularly in identifying and managing offensive content. Offensive content, including hate speech, profanity, and discriminatory remarks, can have severe negative impacts on individuals and communities, leading to harassment, discrimination, and social unrest.

The aim of this project is to develop a robust and efficient system for detecting and classifying offensive language in Turkish text data. To achieve this, we propose a hierarchical encoder transformer model designed for multi-label text classification. The model will first determine whether a piece of text is offensive or not. If the text is identified as offensive, the model will then classify it into one or more specific categories: Sexist, Racist, Profanity, and Insult. This hierarchical approach allows for a more precise and nuanced understanding of offensive content.

1.2 Objectives

The primary objectives of this project are as follows:

- **Develop a Hierarchical Encoder Transformer Model:** To create a model capable of distinguishing between offensive and non-offensive content with high accuracy. This involves fine-tuning pre-trained language models on our specific dataset.
- **Classify Offensive Content into Specific Categories:** To further classify identified offensive content into four distinct categories: Sexist, Racist, Profanity, and Insult. This multi-label classification allows for a more detailed understanding of the nature of the offensive content.
- **Utilize a Preprocessed and Hand-Labeled Dataset:** To leverage a dataset of 81,800 rows that has been meticulously preprocessed and hand-labeled to ensure accuracy and reliability in model training and validation.
- **Improve Content Moderation Tools:** To contribute to the development of better tools and techniques for content moderation on social media platforms, thereby developing a safer and more inclusive online environment.

1.3 Scope

This project focuses on the development and validation of the proposed model using a dataset of Turkish-language content sourced from Twitter and Kaggle. The scope includes:

- **Data Collection:** Compiling a diverse set of text data from various sources to ensure a comprehensive dataset that captures a wide range of language use and offensive content.
- **Data Preprocessing:** Implementing data cleaning, normalization, tokenization, and labeling processes to prepare the dataset for model training.

- **Model Development:** Designing and training the hierarchical encoder transformer model to perform binary classification (offensive vs. non-offensive) followed by multi-label classification for offensive content.
- **Model Evaluation:** Validating the model's performance using various metrics, including accuracy, F1 score, precision, recall, and confusion matrix, to ensure its reliability and effectiveness.
- **Streamlit Demo:** Developing an interactive demo using Streamlit to showcase the model's capabilities and facilitate user interaction with the system.

While the model is designed to be adaptable to various types of textual data, its initial validation will be limited to the dataset at hand. Future work may involve extending the model to other languages and domains, as well as integrating it into real-world content moderation systems.

This report provides a comprehensive overview of the project, including the related work, methodology, experiments and results, discussion of challenges and future work, and conclusions. The following sections will detail each aspect of the project, starting with a review of related work in the field of offensive language detection.

2 Related Work

2.1 Previous Studies on Offensive Language Detection

The detection and classification of offensive language in online content have been the focus of numerous studies over the past decade. Researchers have employed various machine learning and natural language processing (NLP) techniques to tackle this issue. Traditional approaches often relied on rule-based systems and keyword matching [1], which, although useful, lacked the sophistication to handle the nuances of human language.

More recent studies have leveraged advanced machine learning models, including Support Vector Machines (SVM) [2] [3] , Random Forests, and more sophisticated deep learning techniques such as Convolutional Neural Networks (CNN) [4] and Recurrent Neural Networks (RNN) [5]. The introduction of transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers), has significantly improved the performance of text classification tasks, including offensive language detection.

A notable study [6] by Ilyas et al. (2020) introduced a BERT-based model for offensive language detection in Turkish text. Their approach demonstrated the effectiveness of transformer models in understanding context and semantics, leading to higher accuracy in classification tasks. The study utilized a large dataset from social media platforms and applied extensive preprocessing techniques to enhance model performance.

2.2 Differences Between Our Work and Previous Studies

While our work builds upon the foundational studies in offensive language detection, there are several key differences and improvements:

- **Hierarchical Model Structure:** Unlike many previous studies that use a single-stage model for classification, our approach employs a hierarchical model. The first stage determines whether a text is offensive, and the second stage classifies the offensive texts

into specific categories (Sexist, Racist, Profanity, and Insult). This two-step approach allows for more precise and detailed classification.

- **Multi-Label Classification:** Our model is designed to handle multi-label classification, recognizing that offensive texts can belong to more than one category simultaneously. This is a significant enhancement over traditional single-label classification models.
- **Enhanced Preprocessing Techniques:** We implement a comprehensive preprocessing pipeline that includes cleaning, normalization and tokenization. This ensures that the data fed into the model is of high quality, which is crucial for training effective machine learning models.
- **Typo Correction Methods:** In addition to standard preprocessing steps, we explore multiple methods for correcting typographical errors in the dataset. These methods include using the TurkishNLP library, unique words and dictionary matching, and the Jype1 and Zemberek libraries.
- **Interactive Streamlit Demo:** To showcase our model's capabilities and facilitate user interaction, we developed an interactive demo using Streamlit. This allows users to input text and see the model's predictions in real-time, demonstrating the practical applicability of our work.
- **Focus on Turkish Language:** While many studies focus on English-language datasets, our project specifically addresses the detection of offensive language in Turkish. This is particularly important given the linguistic and cultural differences that can affect the interpretation of offensive content.

These enhancements aim to provide a more robust and versatile tool for offensive language detection in Turkish text, addressing the limitations and gaps identified in previous studies.

3 Methodology

3.1 Data Collection

The dataset for this project was compiled from a variety of sources, including publicly available datasets on Kaggle and a collection of tweets. We aimed to gather a diverse set of textual data reflecting a wide range of language use, including different forms of offensive content. The final dataset comprises 81,800 rows of text data, ensuring a rich and comprehensive dataset that supports the nuanced classification tasks required by our project. Each row in the dataset contains a piece of text and its corresponding labels, which indicate whether the text is offensive and, if so, the specific categories of offense (Sexist, Racist, Profanity, and Insult).

3.2 Data Preprocessing

The collected data underwent a comprehensive preprocessing phase to prepare it for use in training the model. This phase included several critical steps to ensure the data's quality and relevance:

3.2.1 Cleaning

Cleaning involved removing any extraneous information that could interfere with the model's performance. This included:

- **Removing URLs and Special Characters:** All URLs, special characters, and non-textual elements were stripped from the dataset to focus solely on the linguistic content.
- **Eliminating Duplicate Texts:** Any duplicate entries were removed to ensure that each piece of text was unique.
- **Filtering Out Short and Meaningless Texts:** Texts with fewer than five characters, or those consisting solely of punctuation, whitespace, or digits, were removed to enhance data quality.

3.2.2 Normalization

Normalization aimed to standardize the text data to a consistent format. This included:

- **Lowercasing:** Converting all text to lowercase to maintain uniformity and reduce the complexity of the text.
- **Correcting Common Misspellings:** Implementing methods to correct frequent spelling errors, ensuring that the text is as accurate as possible.

3.2.3 Tokenization

Tokenization involved breaking down the text into individual words or tokens, making it easier for the model to process and analyze the data. This step is crucial for transforming raw text into a structured format that can be fed into machine learning models.

3.2.4 Labeling

Each piece of text was meticulously annotated with relevant labels using Label Studio. The labels included Not Offensive, Offensive, Sexist, Racist, Profanity, and Insult. This multi-label annotation process enabled precise classification and allowed the model to learn the nuances of different types of offensive content.

In addition to the standard preprocessing steps, we also explored several methods to correct typographical errors in the dataset. These methods included using the TurkishNLP library, unique words and dictionary matching, and the Jype1 and Zemberek libraries. However, due to mixed results and the limitations of these methods, we ultimately decided to proceed with the uncorrected text for model training.

3.3 Model Overview

Our project utilizes a hierarchical encoder transformer model, designed to first identify whether content is offensive and then classify the offensive content into one of four categories. This two-step approach allows for a more thorough understanding and classification of text.

3.3.1 Step 1: Offensive Content Detection

The first step in our model involves detecting whether a given text is offensive or not. This is achieved using a binary classification approach with a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model fine-tuned on our specific dataset. The steps involved in this phase are as follows:

- **Model Selection and Training:** We selected a pre-trained BERT model for its proven effectiveness in understanding context and semantics in text. The model was fine-tuned on our dataset, which included labeled examples of offensive and non-offensive texts. The fine-tuning process involved training the model to optimize for binary classification, with the final layer modified to output probabilities for the two classes.
- **Activation Function:** A softmax activation function was used in the final layer to output a probability distribution for the two classes.
- **Evaluation Metrics:** The model's performance was evaluated using metrics such as accuracy, F1 score, precision, recall, and the Matthews correlation coefficient (MCC). These metrics provided a comprehensive assessment of the model's ability to correctly classify offensive content.

3.3.2 Step 2: Classification of Offensive Content

Once a text is identified as offensive, it is passed to the second stage of our model, which classifies the offensive content into one or more specific categories: Sexist, Racist, Profanity, and Insult. This multi-label classification is performed using a modified BERT model, trained to handle multiple classes simultaneously. The steps involved in this phase are as follows:

- **Model Adaptation and Training:** The same pre-trained BERT model used in the first step was adapted for multi-label classification. The final layer was modified to output probabilities for each of the four offensive categories. The model was then fine-tuned on our dataset, with examples labeled for the presence or absence of each category.
- **Activation Function:** A sigmoid activation function was used in the final layer for each category, allowing the model to output independent probability scores for each class. This approach enables the model to recognize texts that belong to multiple offensive categories.
- **Threshold Setting:** A threshold of 0.5 was applied to the probability scores to determine the final classification. If the score for a category exceeded 0.5, the text was classified as belonging to that category.
- **Evaluation Metrics:** The performance of the multi-label classification model was evaluated using metrics such as accuracy, F1 score, precision, recall, and the Matthews correlation coefficient (MCC) for each category. Additionally, a confusion matrix was used to visualize the model's performance across all categories.

This hierarchical approach, combining binary and multi-label classification, ensures a detailed and accurate identification of offensive content in Turkish text.

4 Experiments and Results

4.1 Data Distribution

Understanding the distribution of labels in our dataset is crucial for evaluating the model's performance and addressing any class imbalance issues. The distribution of the different labels (Not Offensive, Offensive, Sexist, Racist, Profanity, and Insult) is illustrated in the following figures.

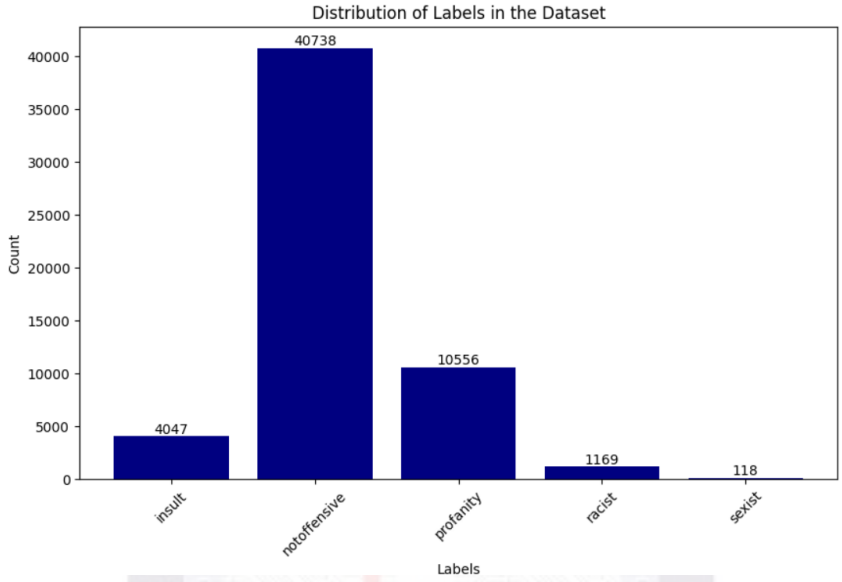

Figure 1: Distribution of Labels in the Dataset

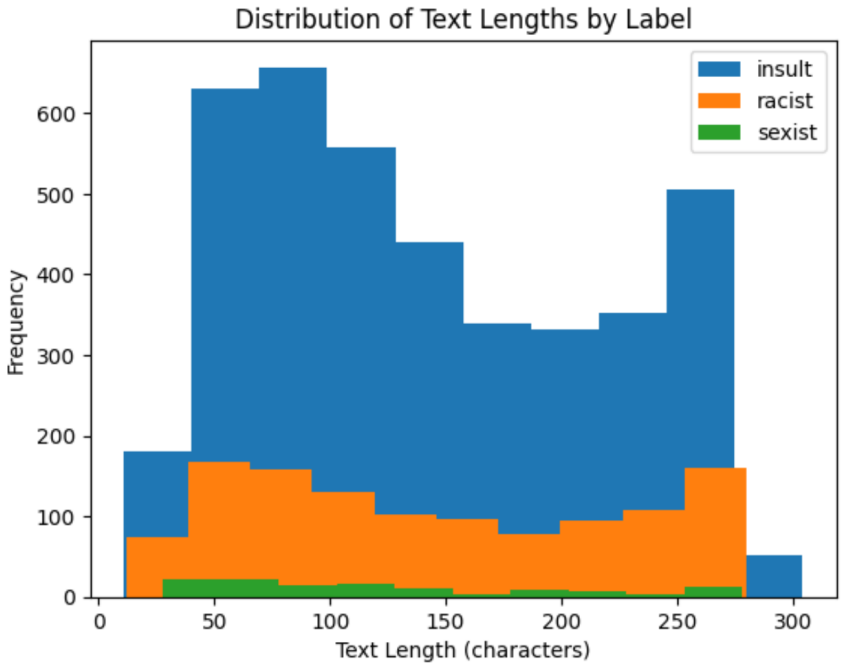

Figure 2: Distribution of Text Lengths by Label: Insult, Racist, and Sexist

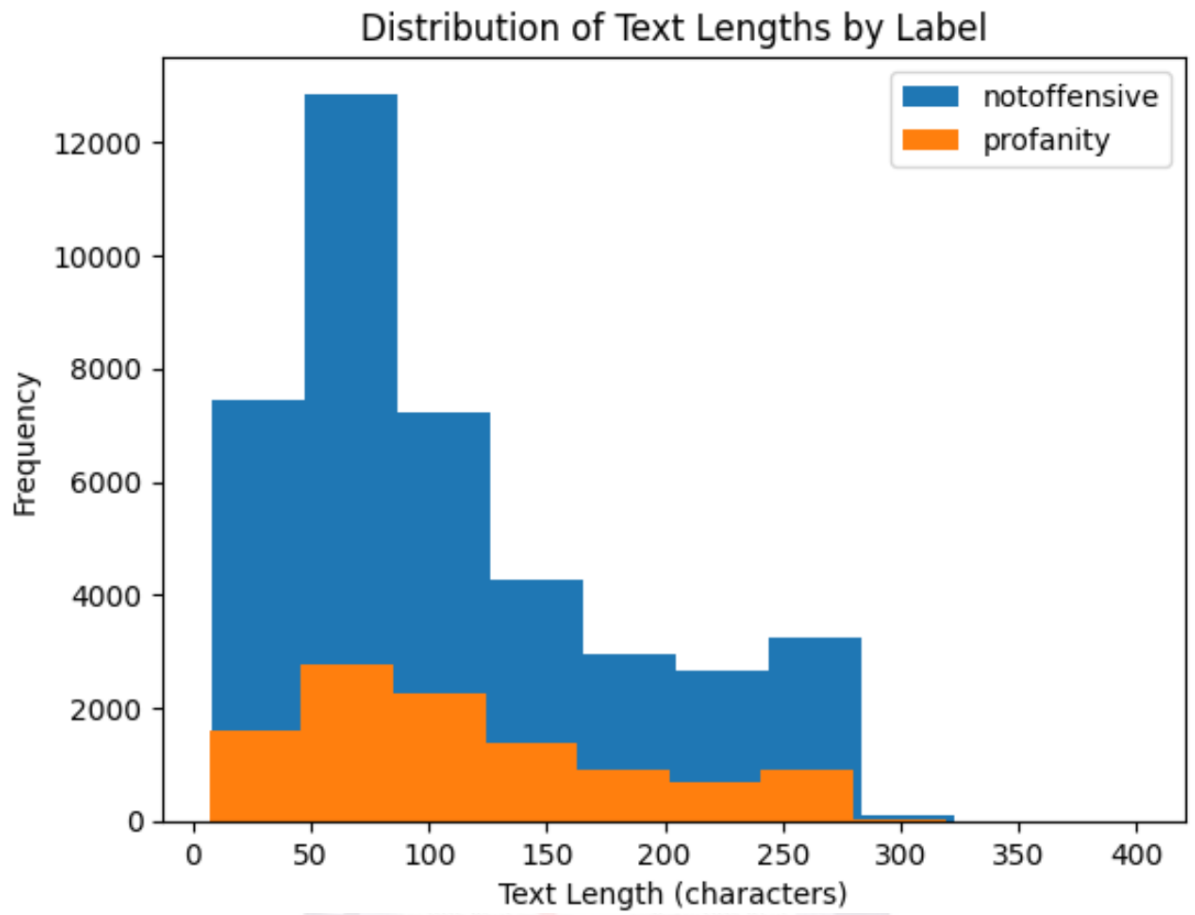

Figure 3: Distribution of Text Lengths by Label: Not Offensive and Profanity

4.2 Typo Correction Attempts

Given the presence of typographical errors in the dataset, we explored multiple methods to correct these errors and improve the quality of the data. Here are the details of the methods attempted:

4.2.1 Method 1: TurkishNLP Library

We first used the TurkishNLP library to correct typographical errors. This library provides tools for spell checking and correction tailored for the Turkish language. Despite its capabilities, the library did not perform as well as expected for our dataset.

```
import turkishnlp
from turkishnlp import detector

obj = detector.TurkishNLP()
obj.download()
obj.create_word_set()

for index, text in enumerate(df['text'].head(10)):
    lwords = obj.list_words(text)
    corrected_text = obj.auto_correct(lwords)
    corrected_text_str = " ".join(corrected_text)
    print(f"Original text #{index+1}: {text}")
```

```
    print(f"Corrected text #{index+1}: {corrected_text_str}")
```

Listing 1: Using TurkishNLP Library

Results:

- **Original text #1:** hemen cep bank yapıyorum ozaman siteye çökücez seninle bugun
- **Corrected text #1:** hemen cep bana yapıyorum ozaman siteye çöküşe seninle bugun
- **Original text #2:** geçmiş olsun fenerin anasini sik
- **Corrected text #2:** geçmiş olsun fenerin anasini sik
- **Original text #9:** demek ki bu piçler yapıyormuş
- **Corrected text #9:** demek ki bu piller yapıyormuş
- **Original text #10:** çünkü evlilik sonu gelmez saçma sapan bir yarıştır
- **Corrected text #10:** çünkü evlilik sonu gelmez saçma sapan bir yarıştır

4.2.2 Method 2: Unique Words and Dictionary Matching

We then explored a method involving unique words extraction and matching them with a Turkish dictionary to find the closest correct word.

```
unique_words = set(word for text in df['text'].head(10) for word in text
    .split())

with open('words.txt', 'r', encoding='utf-8') as file:
    turkish_dict = set(file.read().splitlines())

import difflib
def find_closest_match(word, dictionary):
    matches = difflib.get_close_matches(word, dictionary, n=1, cutoff
    =0.8)
    return matches[0] if matches else word

closest_matches = {word: find_closest_match(word, turkish_dict) for word
    in unique_words}
```

Listing 2: Unique Words and Dictionary Matching

Results: Here are some examples of the corrections made using this method:

- **Original:** buyuruyor, **Closest Match:** buyuru
- **Original:** yet, **Closest Match:** yeti
- **Original:** sonu, **Closest Match:** sonuç
- **Original:** silindi, **Closest Match:** silindir
- **Original:** baksan, **Closest Match:** bakan

Sample Corrected Texts:

- **Original:** hemen cep bank yapıyorum ozaman siteye çökücez seninle bugun

- **Corrected:** hemen cep bank yapıyorum zaman site çökücez sinle bungun
- **Original:** geçmiş olsun fenerin anasini sik
- **Corrected:** geçmiş yosun fenerli anasini sik
- **Original:** migros adet bilet var ilgilenen varsa yazsın
- **Corrected:** migros adet bilet var ilgilenme arsa yazın
- **Original:** çok hızlı gidenlere yavaş demek için geride duruyoruz
- **Corrected:** çok hızlı gidenlere yavaş demek için gerdel duruyoruz

4.2.3 Method 3: Jype1 and Zemberek Library

Finally, we used the Jype1 library to integrate the Zemberek spell-checking library. This method also provided mixed results.

```
from jpype import startJVM, shutdownJVM, JClass, JString

startJVM(getDefaultJVMPath(), "-Djava.class.path=zemberek-tum-2.0.jar")
TurkishMorphology = JClass('zemberek.morphology.TurkishMorphology')
morphology = TurkishMorphology.createWithDefaults()

for index, text in enumerate(df['text'].head(10)):
    analysis = morphology.analyzeSentence(JString(text))
    corrected_text = " ".join([str(result.getBestAnalysis().getLemmas())
    for result in analysis])
    print(f"Original text #{index+1}: {text}")
    print(f"Corrected text #{index+1}: {corrected_text}")

shutdownJVM()
```

Listing 3: Using Jype1 and Zemberek Library

Results:

- **Original text #1:** hemen cep bank yapıyorum ozaman siteye çökücez seninle bugun
- **Corrected text #1:** hemen cep bank yap zaman site çökücez sefine burun
- **Original text #2:** geçmiş olsun fenerin anasini sik
- **Corrected text #2:** geçmiş olgun pelerin anasini sik
- **Original text #9:** demek ki bu piçler yapıyormuş
- **Corrected text #9:** demek kg bu kiler yap
- **Original text #10:** çünkü evlilik sonu gelmez saçma sapan bir yarıştır
- **Corrected text #10:** çinko evlilik son gel saçma sap bir yarış

As can be seen from examples, none of those methods work well for typo correction, so we decided not to use any of the typo correction methods and leave the data as it was before typo correction.

4.3 Model Training and Evaluation

After preprocessing the data, we trained the hierarchical encoder transformer model. The training process and results for each step of the model are detailed below.

4.3.1 Offensive/Not Offensive Classification

The first model was trained to classify texts as offensive or not offensive. We used a pre-trained BERT model fine-tuned on our dataset.

- **Training Settings:** The model was trained for 3 epochs with a batch size of 16, using the AdamW optimizer and a learning rate of 2e-5.
- **Results:** The model achieved a test accuracy of 93.75%, an F1 score of 0.9373, precision of 0.9372, recall of 0.9375, and an MCC of 0.8475.

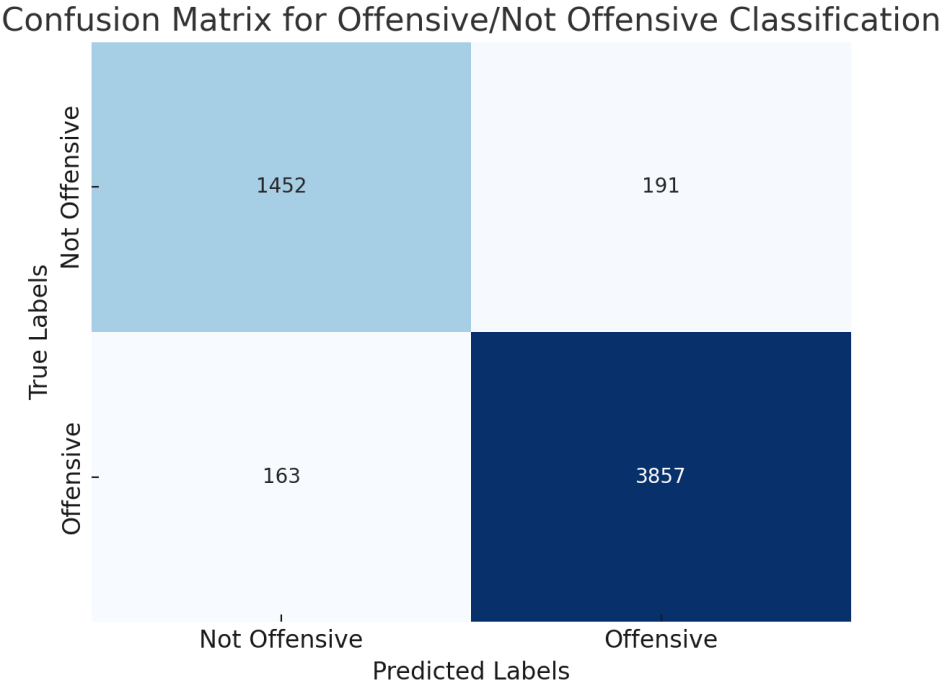

Figure 4: Confusion Matrix for Offensive/Not Offensive Classification

4.3.2 Multi-label Classification

The second model classified offensive texts into four categories: Sexist, Racist, Profanity, and Insult.

- **Training Settings:** Similar settings were used as in the first model, with adjustments for multi-label classification.
- **Results:** The model achieved a test accuracy of 92.27%, an F1 score of 0.8381, precision of 0.8394, recall of 0.8405, and an MCC of 0.8352.

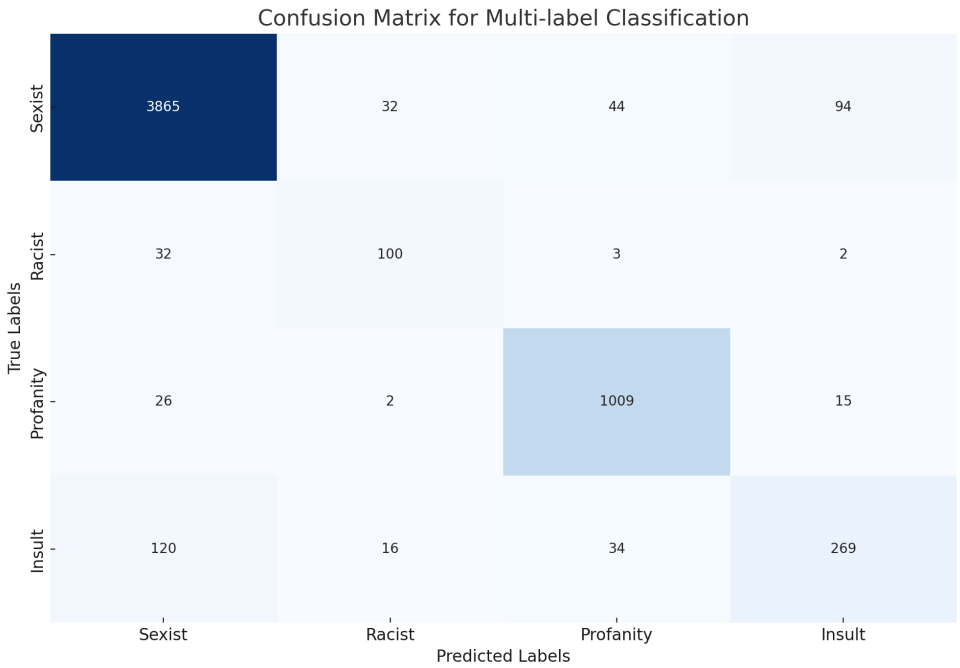

Figure 5: Confusion Matrix for Multi-label Classification

4.4 Streamlit Demo

To showcase our model's capabilities, we developed an interactive demo using Streamlit. This demo allows users to input text and receive real-time predictions from the model, illustrating its practical application.

4.5 Final Results and Analysis

The final results indicate that our hierarchical encoder transformer model performs well in detecting and classifying offensive content in Turkish text. The evaluation metrics demonstrate the model's robustness and reliability. However, there are areas for improvement, particularly in handling typographical errors and expanding the model to other languages and contexts.

5 Discussion

5.1 Challenges Faced

Throughout the course of this project, several challenges were encountered that affected the progress and outcomes. Some of the primary challenges include:

- **Data Quality:** One of the main challenges was ensuring the quality of the data. The presence of typographical errors and inconsistencies in the dataset necessitated extensive preprocessing and typo correction efforts. Despite various methods employed, achieving perfect correction was not possible.
- **Class Imbalance:** The dataset exhibited significant class imbalance, particularly with labels such as 'Sexist' and 'Racist' having considerably fewer instances compared to 'Not Offensive' and 'Profanity'. This imbalance posed a challenge in training the model to accurately classify minority classes.

- **Model Complexity:** The hierarchical model structure, while effective, introduced additional complexity in training and tuning the models. Balancing the performance of both the offensive content detection model and the multi-label classification model required careful consideration and adjustment.
- **Computational Resources:** Training transformer-based models such as BERT is computationally intensive. Limited computational resources and time constraints posed challenges in conducting extensive hyperparameter tuning and experimentation.
- **Language Specificity:** Working with Turkish language text introduced unique challenges due to the language's morphological richness and agglutinative nature. Existing NLP tools and libraries often had limitations in effectively handling Turkish text.

5.2 Improvements and Future Work

Despite these challenges, there are several areas for improvement and future work:

- **Enhanced Typo Correction:** Future work could explore more advanced typo correction methods, potentially leveraging recent advancements in NLP and language models to better handle typographical errors in Turkish text.
- **Balancing Class Distribution:** Techniques such as data augmentation, synthetic data generation, or advanced sampling methods could be employed to address the class imbalance issue and improve model performance on minority classes.
- **Model Optimization:** Further hyperparameter tuning and experimentation with different model architectures could be conducted to optimize the performance of the hierarchical model. Exploring alternative transformer-based models such as RoBERTa or GPT-3 could also be beneficial.
- **Multi-lingual and Multi-domain Expansion:** Extending the model to handle multiple languages and domains would increase its applicability. This would involve training on diverse datasets and incorporating multi-lingual pre-trained models.
- **Real-time Deployment:** Developing a real-time deployment pipeline for the model, integrated with social media platforms or content moderation systems, would demonstrate the practical utility of the project. This could involve optimizing the model for inference speed and scalability.

6 Conclusion

This project aimed to develop a robust and efficient system for detecting and classifying offensive language in Turkish text using a hierarchical encoder transformer model. Through extensive data preprocessing, typo correction efforts, and the implementation of a two-step classification approach, we achieved promising results. The model demonstrated high accuracy in both detecting offensive content and categorizing it into specific types.

Despite the challenges encountered, including data quality issues and class imbalance, the project successfully highlighted the potential of transformer-based models in addressing complex NLP tasks. The insights gained and the methodologies developed provide a solid foundation for future work in this area.

In conclusion, the hierarchical encoder transformer model developed in this project represents a significant step forward in the automated detection and classification of offensive language in Turkish text. With further improvements and expansion, this approach has the potential to be a valuable tool in maintaining safer and more inclusive online environments.

References

[1] Fatemah Husain and Ozlem Uzuner. A survey of offensive language detection for the arabic language. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 20(1), mar 2021.

[2] Gabriel Araújo De Souza and Márjory Da Costa-Abreu. Automatic offensive language detection from twitter data using machine learning and feature selection of metadata. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2020.

[3] Parisa Hajibabaee, Masoud Malekzadeh, Mohsen Ahmadi, Maryam Heidari, Armin Esmaeilzadeh, Reyhaneh Abdolazimi, and James H Jr Jones. Offensive language detection on social media based on text classification. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0092–0098, 2022.

[4] Bharathi Raja Chakravarthi, Manoj Balaji Jagadeeshan, Vasanth Palanikumar, and Ruba Priyadharshini. Offensive language identification in dravidian languages using mpnet and cnn. *International Journal of Information Management Data Insights*, 3(1):100151, 2023.

[5] Anas Ali Khan, M. Hammad Iqbal, Shibli Nisar, Awais Ahmad, and Waseem Iqbal. Offensive language detection for low resource language using deep sequence model. *IEEE Transactions on Computational Social Systems*, pages 1–9, 2023.

[6] Anil Özberk and İlyas Çiçekli. Offensive language detection in turkish tweets with bert models. In *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pages 517–521, 2021.