OFFENSIVE LANGUAGE DETECTION IN TURKISH TWITTER DATA WITH BERT MODELS

BERT MODELLERİYLE TÜRKÇE TWİTTER VERİLERİNDE SALDIRGAN DİL TESPİTİ

ANIL ÖZBERK

PROF. DR. İLYAS ÇİÇEKLİ

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Science

in Computer Engineering

2022

ABSTRACT

OFFENSIVE LANGUAGE DETECTION IN TURKISH TWITTER DATA WITH BERT MODELS

Anıl ÖZBERK

Master of Science, Department of Computer Engineering

Supervisor: Prof. Dr. İlyas ÇİÇEKLİ

January 2022, 50 pages

As insulting statements become more frequent on online platforms, these negative statements create a reaction and disturb the peace of society. Identifying these expressions as early as possible is important to protect the victims. Offensive language detection research has been increasing in recent years. Offensive Language Identification Dataset (OLID) was introduced to facilitate research on this topic. Examples in OLID were retrieved from Twitter and annotated manually. Offensive Language Identification Task comprises three subtasks. In Subtask A, the goal is to discriminate the data as offensive or non-offensive. Data is offensive if it contains insults, threats, or profanity. Five languages datasets, including Turkish, were offered for this task. The other two subtasks focus on categorizing offense types (Subtask B) and targets (Subtask C). The last two subtasks mainly focus on English.

This study explores the effects of the usage of Bidirectional Encoder Representations from Transformers (BERT) models and fine-tuning methods on offensive language detection on Turkish Twitter data. The BERT models that we use are pre-trained in Turkish corpora. Our fine-tuning methods are designed by considering the Turkish language and Twitter data. The importance of the pre-trained BERT model in a downstream task is emphasized. In addition, experiments with classical models are conducted, such as logistic regression, decision tree, random forest, and support vector machine (SVM).

Keywords: Natural Language Processing, Offensive Language Detection, BERT

ÖZET

BERT MODELLERİYLE TÜRKÇE TWİTTER VERİLERİNDE SALDIRGAN DİL TESPİTİ

Anıl ÖZBERK

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Prof. Dr. İlyas ÇİÇEKLİ

Ocak 2022, 50 sayfa

Online platformda hakaret içeren ifadeler arttıkça bu saldırgan ifadeler tepki yaratarak toplumun huzurunu bozmaktadır. Bu ifadelerin erken tespit edilmesi mağdurların korunması açısından önemlidir. Saldırgan dil tespit araştırmaları son yıllarda artmaktadır. Bu konudaki araştırmaları kolaylaştırmak amacıyla Saldırgan Dil Tanımlama Veri Kümesi (OLID) oluşturulmuştur. OLID verileri Twitter'dan toplanmış ve manuel olarak etiketlenmiştir. Saldırgan Dil Tanımlama Görevi üç alt görevden oluşur. Alt Görev A'da amaç, verileri saldırgan veya saldırgan olmayan olarak ayırt etmektir. Hakaret, tehdit veya küfür içeriyorsa veriler saldırgandır. Bu görev için Türkçe dâhil beş dilde veri seti hazırlanmıştır. Diğer iki alt görev, saldırı türlerinin (Alt Görev B) ve hedeflerin (Alt Görev C) sınıflandırılmasına odaklanır. Son iki alt görev için sadece İngilizce veri seti bulunmaktadır.

Bu çalışma, Dönüştürücülerden Çift Yönlü Kodlayıcı Gösterimleri (BERT) modellerinin ve ince ayar tekniklerinin kullanımının Türkçe Twitter verilerinde saldırgan dil tespiti üzerindeki etkilerini araştırmaktadır. Kullandığımız BERT modelleri Türkçe ile ön eğitime tabi tutulmuştur. İnce ayar teknikleri ise Türkçe dili ve Twitter verileri göz önüne alarak hazırlandı. Çalışmamızda önceden eğitilmiş BERT modelin hedef görev üzerindeki önemi vurgulandı. Ayrıca lojistik regresyon, karar ağacı, rastgele orman ve destek vektör makineleri (SVM) gibi klasik modeller kullanılarak deneyler yapıldı.

Anahtar Kelimeler: Doğal Dil İşleme, Saldırgan Dil Tespiti, BERT

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my thesis advisor, Prof. Dr. İlyas ÇİÇEKLİ, for guiding me throughout the research. I am grateful that he gave me freedom in my research and the opportunity to pursue my research interests.

Besides, I would like to thank TÜBİTAK for supporting me in continuing my graduate studies. I wish to express my sincere appreciation to my dear friends and colleagues for their help and good wishes.

Finally, I would like to express my deepest gratitude to my family. They loved me unconditionally and supported my decisions.

TABLE OF CONTENTS

FIGURES

TABLES

SYMBOLS AND ABBREVIATIONS

Abbreviations

BERT	Bidirectional Encoder Representation from Transformer
OLID	Offensive Language Identification Datatset
F1	F1-score
OOV	Out of Vocabulary
MLM	Masked Language Model
NSP	Next Sentence Prediction
SVM	Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency
ELMo	Embeddings from Language Models
CNN	Convolutional Neural Network
RoBERTa	Robustly Optimized BERT Pretraining Approach
TLM	Translation Language Model
ALBERT	A lite BERT
GPT	Generative Pre-Training
SOLID	Semi-Supervised Offensive Language Identification Dataset
LSTM	Long Short-Term Memory
mBERT	multilingual BERT
FPS	Fixed Prefix Stemmer
DistilBERT	Distilled version of BERT
ConvBERT	Improving BERT with Span-based Dynamic Convolution
PTM	Pre-trained Model

INTRODUCTION

Social media usage is rapidly increasing each day, and its impact on human life is also on the rise. Some comments on social media contain offensive expressions that disrupt the psychology of the addressee. Machine learning methods are widely used to detect offensive language instead of rule-based systems. Rule-based methods, such as basic word filters, are not effective because the context of the words, the author of the discourse, and the target addressee are also important features for the utterance to be classified as offensive [1]. In recent years, interest in offensive language detection has increased. In addition to research on offensive language detection, some previous studies have focused on different aspects of offensive language detection, such as hate speech detection [2, 3], abusive language detection [4], and bullying detection [5]. These concepts are related, but they are slightly distinct [6]. For example, cyberbullying is an aggressive behavior against a victim; offensive language is more general, does not target a specific person. The surveys [6, 7] present a systematic review of definitions and sources related to these abusive subjects.

OffensEval 2019 task [8] was organized for offensive language identification for English. Three subtasks were created regarding offensive language identification (Subtask A), offense type categorization (Subtask B), and offense target identification (Subtask C). The subject of all subtasks is assigning the test dataset to predefined labels. Multilingual datasets on different languages were included in offensive language detection in OffensEval 2020 task [9]. Both tasks aroused a lot of interest, and many offensive language detection systems competed on the given tasks. One of the datasets in OffensEval 2020 task is the offensive language identification dataset on Turkish tweets. We have tested some pre-trained BERT [10] models on this Turkish dataset. Our BERT-based models obtained compatible results with the best systems on the Turkish dataset [11]. Turkish dataset was collected from Twitter and manually annotated by volunteers.

BERT is a pre-trained language model that yields state-of-the-art results in many NLP tasks. After the model is pre-trained with a large dataset, it is fine-tuned with the target dataset. While the pre-training phase helps the model understand the language, the fine-tuning process helps the model learn features for the target task. BERT can process the words in a sentence in parallel utilizing transformers. Processing words in parallel rather than sequentially allows the model to learn the context information of a word based on its surroundings. Different BERT models could be generated by changing the pre-training dataset or the architecture.

In this study, we research the effect of different BERT-based models and fine-tuning methods on Turkish offensive language classification. We used BERT models that are trained on Turkish texts such as BERTurk, DistilBERTurk, and ConvBERTurk [12]. We refer to operations that we perform on the dataset during the fine-tuning phase by fine-tuning methods. We designed our fine-tuning methods considering twitter specific features such as hashtags and emojis and language-specific features such as average word length in the Turkish language. We also pay attention to word statistics to remove or add dictionary definitions in fine-tuning strategies. Our empirical results show that the significant effect on the performance of the system is the used BERT model. Fine-tuning methods have a minor impact on the performance of the system.

Contributions

- We made a comprehensive analysis of fine-tuning methods and BERT models for Turkish offensive language detection.

- We observed that the pre-trained BERT model is more substantial on performance than fine-tuning methods. Data-specific features may seem promising, but they did not affect the results as much as different BERT models.

- We performed an extensive literature review on Offensive Language Identification in Social Media in the context of OffensEval 2019 and OffensEval 2020.

The Outline of the Thesis

- Chapter 0 presents background information about text classification, classical models, pre-trained language models, transfer learning, and BERT models.
- Chapter 3 presents the related works on offensive language identification tasks and explains the dataset.
- Chapter 4 details our fine-tuning methods, pre-trained BERT models, and experimental setup.
- Chapter 5 presents and analyses the results of the models.
- Chapter 0 concludes the thesis with a summary.

BACKGROUND

In this chapter, we will briefly mention the text classification problem, the classical classification methods that we used, and the importance of deep networks in the classification problem. Since the BERT model is crucial in our research, we will give information about the BERT model, pre-trained language models, and transfer learning.

Text Classification

Text classification is a classical natural language processing (NLP) problem that tries to assign predefined labels to text units such as sentences, paragraphs, documents, etc. The text classification process involves the following steps:

- **Text preprocessing techniques** aim to remove noise in a text and make the text more valuable for NLP models. Stemming, converting text to lower or upper case, removing punctuations, removing stop words are typical preprocessing techniques. The nature of a given text and target application must be taken into consideration when applying preprocessing steps. For example, stemming has little influence in Turkish text classification when the dataset is large [13], in English and Czech classification task has a negative influence [14], in German information retrieval has a positive effect [15].

- **Text representation or transformation**: Most classification algorithms take inputs as vectors or matrices. Term frequency - Inverse document frequency (TF-IDF) is a method that represents a text in vector format while considering the importance of words. The formula of TF-IDF is shown in Eq. 1. $tf_{t,d}$ and idf_t refer to term frequency and inverse document frequency of *t*, respectively. Word embedding and transformers are other choices for text representation.

$$TF\text{-}IDF_{t,d} = tf_{t,d}.idf_t \tag{1}$$

- **Selection and application of classification techniques**: Most classical methods contain two parts. In the first part, features are extracted from the text by the methods such as bag of words. In the second part, selected features are fed into classifiers. Two phases methods have some limitations. For instance, these methods need feature engineering which is mostly tedious, and extracted features are not generic for every classification task. To address the limitation of using hand-crafted features neural approaches have been explored. The core component of these approaches is a machine-learned embedded model that maps the text to low-dimension continuous feature vectors. In this way, the need for feature extraction is eliminated.

- **Classifier evaluation:** The confusion matrix is shown in Table 0.1. TP means True Positive, FP means False Positive, FN means False Negative, TN means True Negative.
  For the evaluation of results, recall, precision, and F1 values are taken into consideration. Recall, precision and F1 are calculated as shown in Eq. 2, 3, 4, and 5.

Table 0.1 Confusion Matrix

Predicted class / **Actual class**	**N**	**P**
N	TN	FP
P	FN	TP

Recall shows the proportion of actual positives that are identified correctly.

$$Recall = \frac{TP}{TP + FN} \qquad (2)$$

Precision shows the proportion of positive identifications that are actually correct.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

F1 score is the harmonic mean of Recall ad Precision.

$$F1 = \frac{Recall * Precision}{Recall + Precision} \tag{4}$$

Macro-F1 score is the arithmetic mean of individual per-class F1-scores where C is the total number of classes:

$$macro - F1 = \frac{1}{C}\sum_{i=1}^{C} F1_i \tag{5}$$

Classical Methods

Machine learning methods intend to predict the outcome of new observations based on previous observations. Machine learning algorithms try to perform these operations without being explicitly programmed. In this thesis context, the classical methods refer to machine learning methods that use algorithmic structures to solve the problem. Experts select the algorithm and features for these methods.

Logistic Regression

Logistic Regression is a binary classifier, and it tries to predict the hyperplane that separates two classes in the best way [16]. Logistic function, also called sigmoid, is the core of Logistic Regression. The logistic function maps prediction probabilities between 0 and 1. The probability of the label being y_i is determined by Eq. 6, x_n are training samples, β_n are parameters that try to be learned by maximum likelihood.

$$P(y_i) = \frac{1}{1 + e^{-(\beta_0 + \cdots + \beta_0 X_0)}} \tag{6}$$

Decision Tree

Decision trees use a tree-like structure decision mechanism to classify the dataset [17]. Each node contains a decision mechanism. According to decisions made by the root node, the decision algorithm proceeds to the next decision node. Leaf nodes are the last stage for classifying data into class labels. Decision trees are easy to construct and understand; however, they suffer from low bias and high variance. Removing some parts of the decision tree by pruning or bootstrapping multiple decision trees are enhancement methods for overfitting.

Random Forest

Random forest [18] is an ensembling algorithm that combines the results of more than one algorithm to solve problems. Different trees are constructed using different bootstrap samples of the data. While standard trees choose the best variable to split each node, Random Forest chooses the split variable from a subset of possible variables. Outputs of trees are ranked then the highest one is selected for prediction. It reduces the overfitting in decision trees by improvements in bootstrapping. Figure 0.1 illustrates a simple overview of Decision Tree and Random Forest.

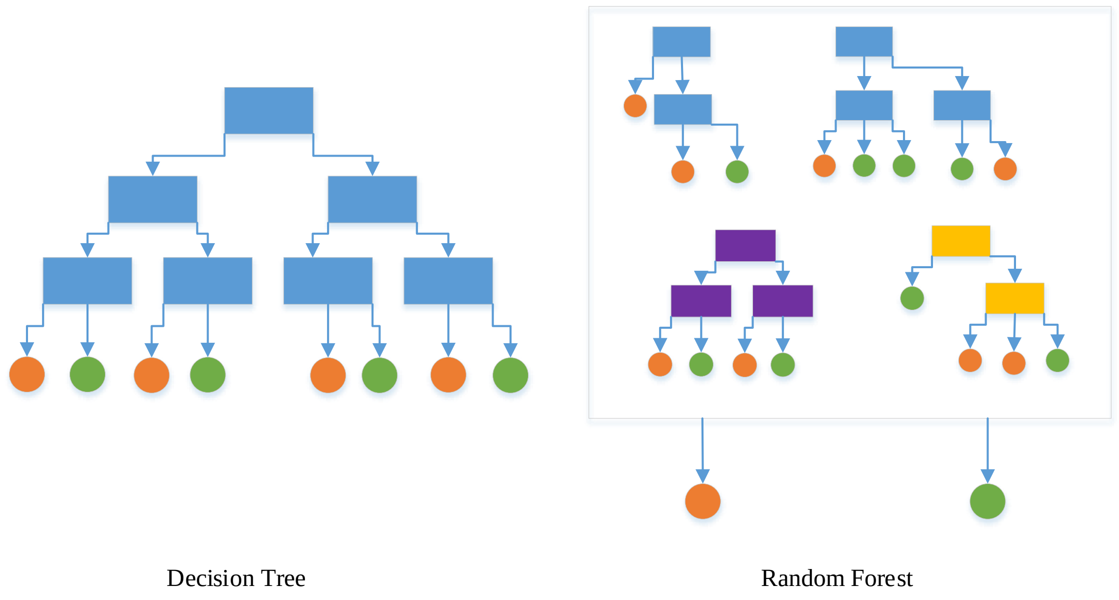

Figure 0.1 Decision Tree and Random Forest

SVM

SVM [19] is a supervised learning model used for regression and classification tasks. Finding hyperplanes between data clusters is the primary goal of SVM. Maximum margins are found to detect optimum hyperplanes. Support vectors are the closest data points to hyperplanes in each data cluster. An example visualization of a possible hyperplane for a linearly separable dataset is given in Figure 0.2.

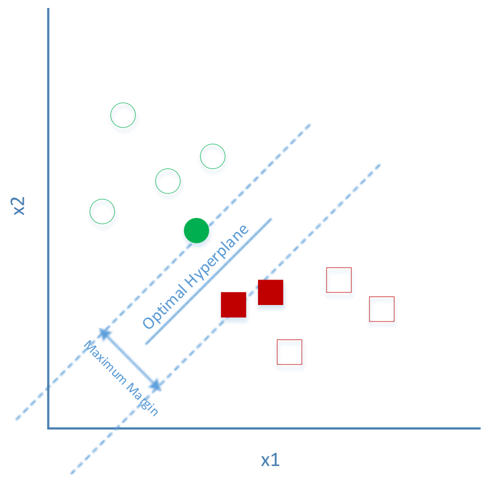

Figure 0.2 SVM Hyperplane

$$\max \gamma(\omega, b) \quad such\ that\ \forall_i\ \ y_i(\omega^T x_i + b) \geq\ 0 \tag{7}$$

Eq. 7 gives intuition of the algorithm. It suggests maximizing the margin between each data point and hyperplane. γ is the margin, $(\omega^T x_i + b)$ is the support vector, y is the class of prediction.

Pre-trained Language Models

Pre-trained language models [20] are trained with task-independent large datasets. In this manner, the model understands the language before data is used for the downstream task. Pre-trained models drew attention in 2006 [21]. Imagenet [22] shows the success of pre-trained models in computer vision. Training the model on unlabeled images resulted in the more accurate tuning of the weights in the deep network. The model trained with a large set of images was used with fine-tuning for smaller tasks. In this chapter, we summarize key concepts that lead to pre-trained language models and discuss pre-trained models from NLP perspective.

In order to process the words in a text, it is necessary to obtain vector representations. Two different word embedding methods are used: non-contextual and contextual. Non-contextual methods such as Word2vec [23], GloVe [24] do not regard the context of the word. These methods cannot capture high-level concepts such as polysemy due to the embedding of words being static. Another problem is that they cannot represent out of vocabulary (OOV) words. Character-based embedding methods such as CharCNN [25] and fastText [26] had developed to compensate for this deficiency.

In the contextual embedding method, the context is considered when creating the word vector. Thus, context-dependent features are obtained. ELMo [27] concatenates the representations of two-layer Bidirectional LSTM that are pre-trained on a bidirectional language modeling task. Each layer contains forward and backward passes. The ULMFiT [28] model expands pre-trained language models by adding a fine-tuning step for classification tasks. The ULMFiT uses Bidirectional LSTM architecture. Recent pre-trained language models use more advanced architectures such as Transformer. OpenAI GPT [29] utilizes a decoder stack for generating text in an autoregressive manner. Autoregression allows the output of a sequence to be the input for the next step. BERT loses autoregression to gain the ability to be bidirectional. BERT splits data into fixed-length segments, and this method does not respect semantic features such as sentence boundary. Hence, the model fails to capture long-term dependency beyond the specified length. Transformer-XL [30] passes previously processed segment information to the current segment. Thus fragments are recurrently connected with each other, and the model can capture long-term dependencies. It is found that BERT was undertrained. To improve the training procedure, RoBERTa [31] experimented with methods; training the model longer using a larger dataset, removing NSP objective, increasing sequence size, and changing masking pattern during training time. Importantly, RoBERTa uses extra datasets for training. RoBERTa outperforms BERT and XLNet [32] on most GLUE benchmark results. The proliferation of non-English models has led to the idea of creating models that can work on any language. To reduce the parameter size and communication cost of BERT, ALBERT [33] proposes two methods: factorized embedding parameterization and cross-layer parameter sharing. Although ALBERT models have fewer parameters than corresponding BERT structures, the training and inference phases of ALBERT models take longer. Google research released mBERT, which is a multilingual version of BERT. mBERT [10] is pre-trained with a shared vocabulary and capable of working with 104 top languages. XLM [34] improves BERT performance by translation language model (TLM). TLM applies MLM objective on two bilingual sentences which have the same meaning. XLM could leverage the representation of low-resource languages by using high-resource languages counterparts. XLM-R [35] is a version of RoBERTa trained with large multilingual corpora, 2.5TB CommonCrawl data. The model outperforms previous multilingual language models on cross-lingual benchmarks.

Transfer Learning

Transfer learning [36] is a methodology that allows the features learned by a model to be used in other models that have similar domains. Adapting pre-trained language models to downstream tasks is a transfer learning method. There are a few issues to consider when transferring information to downstream tasks.

- A pre-trained model should be selected that is compatible with the needs of the task to be used. For example, while BERT model achieves high accuracy in question answering tasks due to NSP, it is not suitable for the sentence generation task.
- The data of the pre-trained models and the data of the downstream task should be similar. For example, the English pre-trained model - English downstream task will achieve more accurate results than the Turkish pre-trained model - English downstream task. There are a large number of pre-trained models (PTMs) that satisfy the need for different languages and domains.
- Whether to fine-tune the model for the downstream task should be decided. BERT fine-tune the parameters to adjust the parameters according to the downstream task. ELMo utilizes a feature-based approach to enable the use of predetermined parameters for each downstream task.

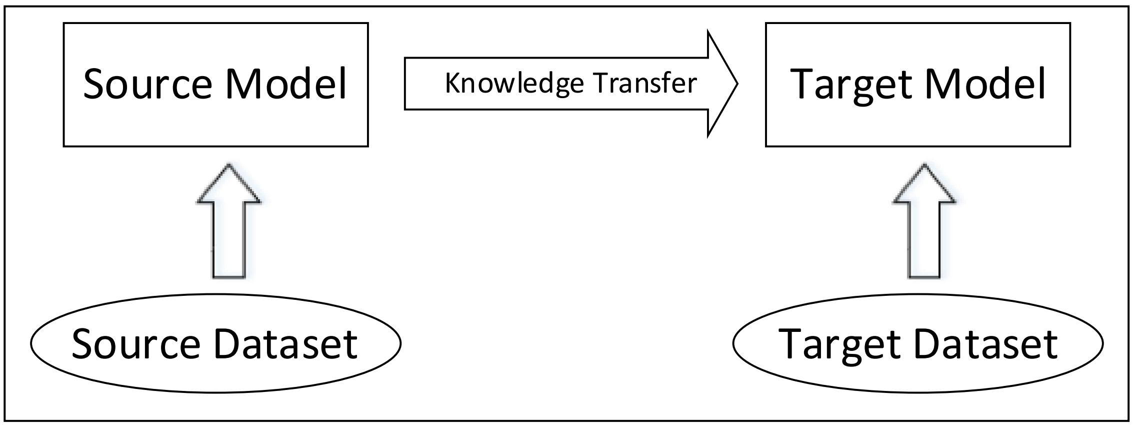

Figure 0.3 Transfer Learning, adapted from [20]

Transformer Architecture

Transformer [37] architecture that uses attention mechanisms [38] was proposed as an alternative to Long Short-Term Memory networks (LSTM) and other Recurrent Neural Networks models. The attention mechanism helps the model focus on other words and decides which focused words are important. For example, in the sentence “The dog didn’t cross the street because it was too tired” it is hard to decode the meaning of the word “it” for LSTM networks because LSTM networks process sentences unidirectional, and these networks have a long-term dependency problem. On the other hand, transformers could understand the “it” and “dog” relation because of the attention mechanism. Self-attention solves the problem by updating hidden states while looking at other words in the entire sequence. Figure 2.4 shows the attention pattern produced by the selected attention head in the encoder stack’s last layer. The left half of the figure shows that the word “dog” is among the words that the selected attention head pays attention to while processing the word “it”. Similarly, the right half of the figure shows the importance of the context of the word “it” for the word “dog”.

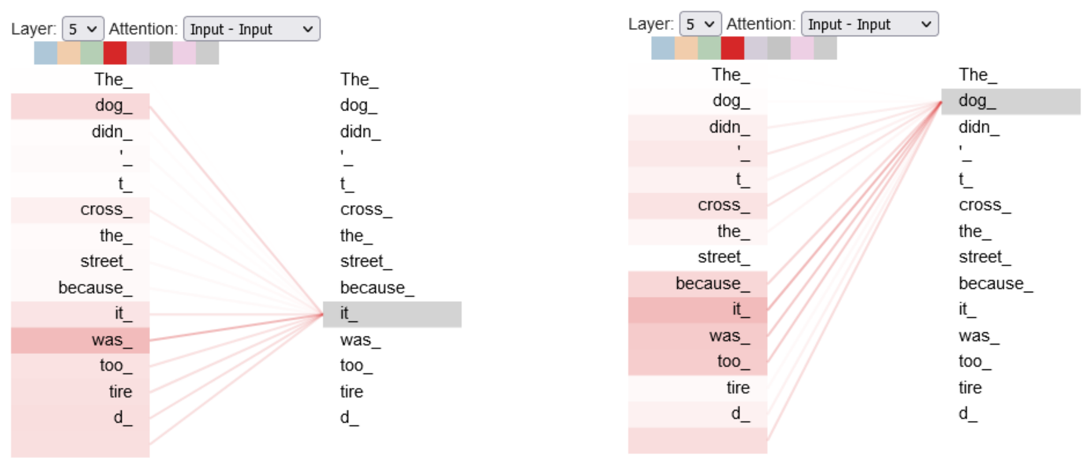

Figure 0.4 Attention Visualization (Tensor2Tensor [39] library)

Figure 2.5 shows the illustration of the Transformer model. It contains encoder and decoder stacks. While encoders create representations of inputs, decoders construct outputs. Self-attention blocks convert tokens to key (K), query (Q), and value (V) matrices by applying three linear transformations as shown in Eq. 8. X denotes given inputs, W indicates trainable weight matrices. The dot product of K and Q generates the attention score, V represents the original token. Attention score calculated after division with the dimension of key vector as shown in Eq. 9.

$$V = W^V X,\ K = W^k X,\ Q = W^q X \tag{8}$$

$$Attention(Q,K,V) = softmax(\frac{QK^T}{d_k})V \tag{9}$$

Lample and Conneau [33] extend attention with multi-head attention (Eq. 10), which combines attention calculations for each split of K, Q, and V matrices. Combining attention heads allows the model to find different patterns in the input sequence. Lastly, to represent the order of the words, a positional encoding vector is given to the model.

$$MultiHead(Q,K,V) = Concat(head_1, \ldots.., head_n)W_O \tag{10}$$

$$head_i = Attention(QW^i, KW^i, VW^i)$$

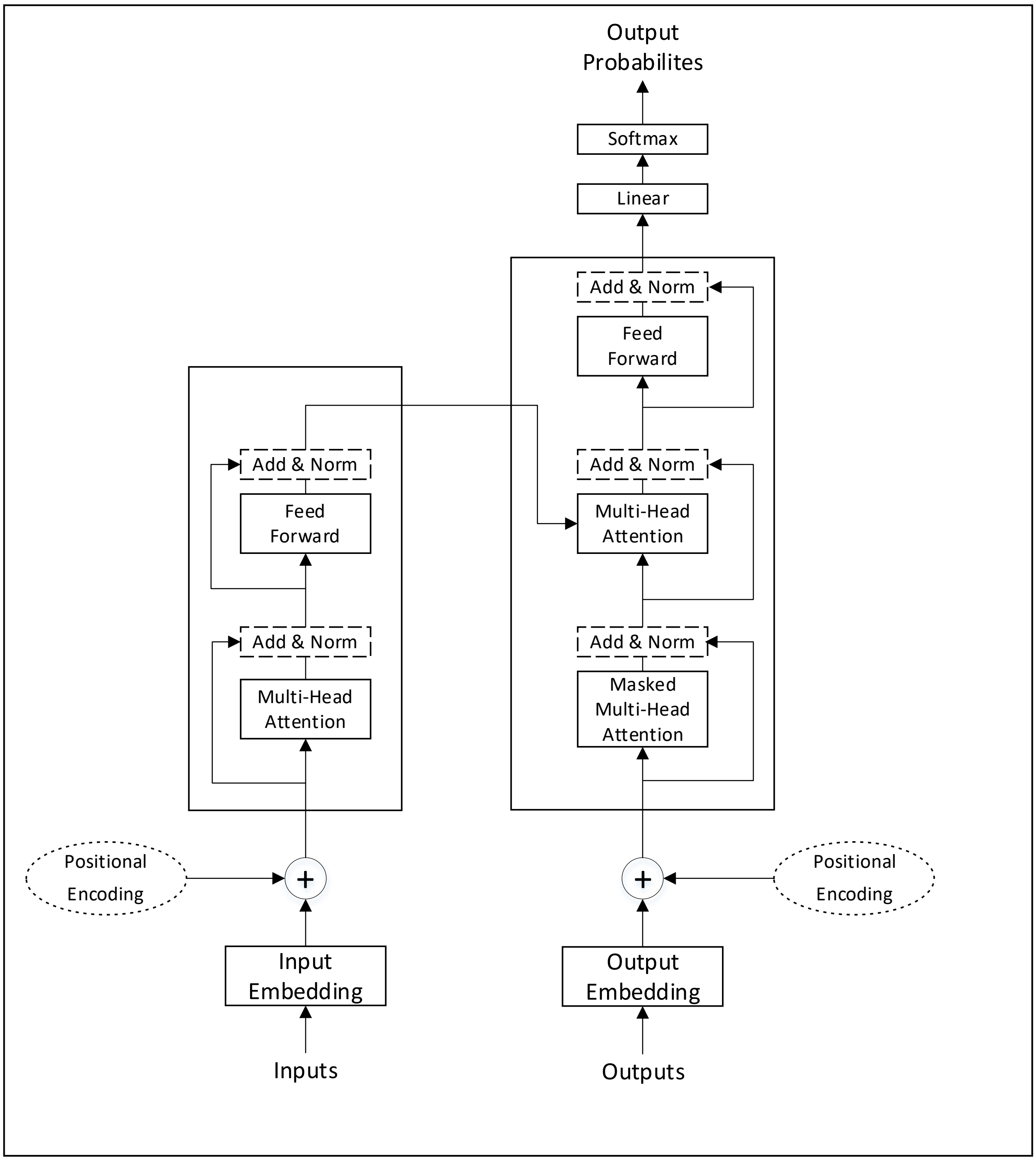

Figure 0.5 Transformer Model, adapted from [37]

BERT

BERT pre-trains a bidirectional deep neural network that uses transformer architecture. BERT was pre-trained with the masked language modeling (MLM) and next sentence prediction (NSP) objectives. MLM is simply a filling in the blanks task [40]. BERT randomly chooses a word from a sentence and tries to predict that word from the context. While this masking word estimation process, BERT tries to predict the word by combining the right and left context in which the word occurs. NSP is a classification algorithm that tries to find the successor and premise of the given two sentences. The BERT base model contains 12 layers of 768 hidden size and 12 self-attention heads. The BERT-large model contains 24 layers of 1024 hidden size and 16 self-attention heads.

Figure 2.6 shows simple classification processes. Encoder stack processes sentence tokens in parallel. An encoder contains a self-attention block and a feed-forward neural network. After calculating token representations with the encoder, these representations are fed into an output layer for the classification task. [CLS] is a special token that indicates this task is a classification task. [SEP] token indicates the end of the sentence. BERT operates on the fixed-length input. Sentences shorter than the predefined length have to be padded with empty tokens. The original BERT model's fixed input size is 512.

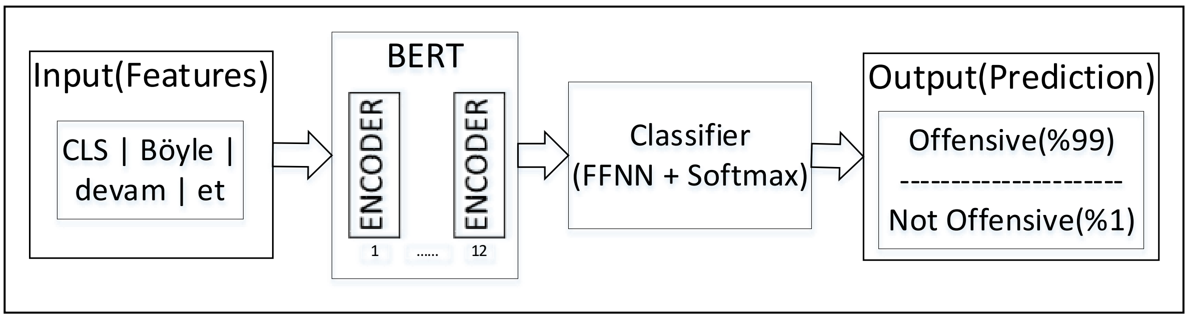

Figure 0.6 BERT Model Classification

Bert Tokenization and Encoding

Tokenization is a process to divide the text into words and sub words. BERT model assigned unique IDs to each token. Calculated tokens need to be converted its corresponding IDs. When pre-trained models are used, some words in the test data might not appear in the model's dictionary. This problem is called OOV problem. The symbol [UNK] could be used to replace OOV words, but this leads to information loss. To overcome of OOV words representation problem, BERT uses Wordpiece [41] tokenization algorithm. The process is started with creating a language that contains individual characters in the language, then the most frequent symbol combinations in the vocabulary are iteratively added to the new vocabulary. The likelihood of this new symbol must be greater than the previously added symbol for adding the dictionary. A sentence’s journey throughout the tokenization and encoding process is shown below.

Original Sentence: Tembellik ve uyku düşmanımızdır.

Tokenized Sentence: ['Tem', '##bel', '##lik', 've', 'uyku', 'düşmanı', '##mız', '##dır', '.']

[CLS] and [SEP] Added Tokens: ['[CLS]', 'Tem', '##bel', '##lik', 've', 'uyku', 'düşmanı', '##mız', '##dır', '.', '[SEP]']

Padded Tokens: ['[CLS]', 'Tem', '##bel', '##lik', 've', 'uyku', 'düşmanı', '##mız', '##dır', '.' , '[SEP]', '[PAD]']

Token IDs: [2, 3403, 10197, 2102, 1992, 8055, 16359, 4666, 2077, 18, 3, 0]

RELATED WORKS

Offensive Language Identification

Offensive language detection is a text classification problem, and it has received a lot of attention in recent years. There are related but slightly different topics in literature, such as hate speech [2, 3], abusive content [4], and bullying [5]. Several workshops for offensive content such as ALW2 [42] and TRAC [43, 44] were created. Supervised learning is often used to classify offensive language. The dataset is annotated according to whether it is offensive or not, and the model is expected to predict the label of the test data. Therefore, datasets are critical in training and testing the model. Most of the adversarial attacks work caused performance degradation, and hence they show weaknesses of machine learning methods for hate speech detection [1].

In 2019 and 2020, OffensEval tasks [8, 9] were organized for offensive language identification, and many systems competed in the given tasks. For OffensEval 2019 task, a dataset [45] was created for English, which consists set of tweets that are tagged as Offensive and Not Offensive for Subtask A; for Subtask B and Subtask C, other annotation schemas were utilized, which we will briefly mention. Turkish dataset was included in OffensEval 2020 [9] created for Subtask A. We used this Turkish dataset [11] introduced for the offensive language identification task in our experiments. Our BERT-based models achieved performance as efficient as the performance of the top system in OffenseEval-2020. In the remainder of this chapter, we provide information about the top systems of OffensEval 2019 and 2020. We focused on the results of Subtask A.

Pre-trained transformer models achieve high performance in offensive language detection. Lots of the models [46-50] competed in OffenseEval-2019. The participating teams that applied pre-processing steps to BERT models stood at the top [46, 49, 50]. BERT models achieve top results when compared to the ensemble of a large variety of different models [47]. The ensemble model had overfitted the training data. NLPR@ SRPOL [48] combines several models: LSTM, Transformer, OpenAI's GPT, Random forest, SVM in an ensemble with various embeddings: ELMo, fastText, custom embeddings, and Universal Sentence Encoder. They also used an openly available dataset and custom-built corpora labeled by linguists.

Pre-trained models in the target language are effective in identifying offensive languages [46, 51-53]. LT@Helsinki [52] ranked 1st in Danish Subtask A using NordicBERT. They applied some pre-processing techniques, such as reducing the length of characters that appear more than two consecutive times and segmenting hashtags into words by adding white spaces before every capital letter. LISAC FSDM-USMBA [53] stood at the 1st rank for Arabic. They used BERT to create tweets representations and fed tweet representations to a sigmoid classifier. They also translated emojis into Arabic meanings. SU-NLP [54] and KUISAIL [55] managed to rank in the top three in Subtask A for the Turkish language by utilizing BERTurk. NTU_NLP [56] intends to take advantage of the hierarchical structure of the English dataset. Their model solves all three subtasks simultaneously. The architecture has three layers. The output of each layer is used to form the input of the next layer. The input of the first layer is the output of BERT. The input of the second layer is the concatenation of the output of the first layer (D1-OUT) and the output of BERT. The input of the third layer is the concatenation of the output of the second layer (D2-OUT) and the output of BERT. D1-OUT is directly connected to the output layer of Subtask A, D2-OUT is directly connected to the output layer of Subtask B, the output of the second layer OUT is directly connected to the output layer of Subtask C.

Multilingual models gained popularity with the introduction of multilingual datasets in 2020. When comparing monolingual and multilingual language models on offensive language detection, it is found that monolingual models achieve higher F1 scores [51, 57]. Galileo [58] obtains the highest F1 score in the Turkish language (0.8258) using an ensemble of XLM-R base and XLM-R [35] large models, followed by multilingual fine-tuning using the OLID and SOLID except for English. Rouges [59] fine-tuned XLM-R with all five languages in sequential order: English, Turkish, Greek, Arabic, and Danish. NLPDove [60] used semi-supervised labels from SOLID [9] and cross-lingual language with data selection. Moreover, they used an ensemble of several mBERT models that use minor variations in parameters. GruPaTo [61] retrained mBERT model with MLM objective using language-specific Twitter messages along with fine-tuning the model with language-specific training sets. I2C [62] fine-tunes the mBERT model using the dataset generated by the Smote-Tomek method and the original dataset. LIIR [63] used mBERT and applied a cross-lingual augmentation approach to enhance the training dataset. ALT [64] participated with an ensemble of SVM, DNN, and mBERT models for Arabic and mBERT models for English.

Several works [54, 58, 60, 65-67] focused on ensembles of pre-trained transformer models. UHH-LT [65] used an ensemble combining four different ALBERT models (large-v1, large-v2, xxlarge-v1, and xxlarge-v2). This team achieved the best performance in English Subtask A in OffenseEval-2020. Using Convolutional Neural Network (CNN) with LSTM, Bidirectional LSTM with attention besides BERT to combine into an ensemble SU-NLP makes a slight improvement over BERT model. Moreover, researchers of SU-NLP included the training data they collected for the LSTM model training. Some researchers used BERT to obtain contextualized word vectors [55] and used CNN for classification. The team amsqr [68] utilized a stacked ensemble of neural networks.

Experiments [46, 52, 66] show that machine learning models cannot surpass the performance of pre-trained language models. In 2019 most successful non-BERT model is MIDAS [69] which ranked 5th in Subtask A. They used an ensemble of CNN, Bidirectional LSTM with attention, and Bidirectional LSTM + Bidirectional GRU. In 2020 most participants that used machine learning models combine them with other pre-trained models to create an ensemble. JCT [70] ranked 6th in Danish using random forest. INGEOTEC [71] produces decision function values for the same dataset then combines these values in a classifier.

Pre-processing methods were widely utilized in OffensEval tasks. Substituting emoji to equivalent text [46, 72-75], converting hashtags to text [55, 64, 74, 75], text lowercasing, [49, 54, 72, 73], and removing punctuations [54, 64] are common pre-processing methods. Removing symbols and emoji substitution degrades performance in the English language [72]. To the best of our knowledge, no study has been conducted to examine pre-processing methods on performance in detail.

Several studies were conducted to solve the class imbalance problem by increasing the size of the dataset. AlexU-BackTranslation-TL [76] augment the English dataset by translating tweets to another language then translating to English. By applying this process, they had new versions of existing tweets. Ferryman [67] used the cross-lingual approaches that translates a sentence to three other languages then adds translations to the training dataset. NLPR@ SRPOL, KS@LTH (for English), TysonYU [74], amsqr utilized external datasets. ALT showed that external data is useful for performance improvement. SU-NLP downsample non-offensive dataset to eliminate risks of the data imbalance problem.

Some words used in Twitter are not encountered in books or newspapers. The vector of these words is problematic. The character-level encoding used to avoid this problem worked well to detect the offensive language in Arabic [77]. In addition, character gram features are more helpful than word gram features in random forest [70]. CharacterBERT [78] is a BERT version that uses Character-CNN module to represent wordpiece tokens. CharacterBERT achieves either the same or better results in classification problems than BERT, and it is more robust to misspellings.

Table 0.1 summarizes the performance of the teams for the described classification schema. PTM column indicates if the model used a pre-trained model; ML-PTM column indicates if the model used the multilingual pre-trained models; Ensemble column indicates if the model used an ensemble of models; Pre-processing column indicates if the model applies pre-processing methods; Dataset Change column indicates if the model used dataset other than provided by OffensEval organizers. Dataset change also includes the usage of SOLID and data augmentation methods.

Table 0.1 Overview of Team Performances in Subtask A

Team	PTM	ML-PTM	Ensemble	Pre-processing	Dataset Change
NULI [46]	✓			✓	
Nikolov-Radivchev [47]	✓		✓	✓	
NLPR@ SRPOL [48]	✓		✓	✓	✓
UM-IU@ LING [49]	✓			✓	✓
Embeddia [50]	✓			✓	
KS@ LTH [51]	✓			✓	✓
LT@Helsinki [52]	✓			✓	
LISAC FSDM-USMBA [53]	✓			✓	
SU-NLP [54]	✓		✓	✓	✓
Kuisail [55]	✓			✓	
NTU_NLP [56]	✓			✓	✓
ANDES [57]		✓			
Galileo [58]		✓	✓		
Rouges [59]		✓			
NLPDove [60]		✓	✓	✓	✓
GruPaTo [61]		✓			✓
I2C [62]		✓		✓	✓
LIIR [63]		✓			✓
ALT [64]		✓		✓	✓

UHH-LT [65]	✓		✓		
Guir [66]	✓		✓	✓	
Ferryman [67]	✓		✓	✓	
amsqr [68]	✓		✓		✓
MIDAS [69]			✓	✓	
JCT [70]			✓	✓	
INGEOTEC [71]			✓	✓	✓
BNU-HKBU [72]	✓			✓	
YNUWB [73]				✓	
TysonYU [74]	✓			✓	✓
Kungfupanda [75]	✓		✓	✓	
AlexU-BackTranslation-TL [76]	✓		✓	✓	✓

Dataset

The dataset in OffenseEval-2019 is OLID which consists of three parts. In Subtask A, models need to identify if a given tweet is offensive or non-offensive. In Subtask B, the goal is to categorize offensive tweets as targeted or untargeted. In Subtask C, the focus is to detect the target type in an offensive tweet. Offensive tweets in Subtask A are the dataset of Subtask B; targeted tweets in Subtask B are the dataset of Subtask C. OffensEval 2020 does not provide an additional hierarchical dataset for English. A large weekly labeled Semi-Supervised Offensive Language Identification Dataset (SOLID) for English is published. In addition, for Subtask A, multilingual datasets for languages: English, Turkish, Greek, Arabic, and Danish are included.

We focused on Subtask A, known as the offensive language identification task. In this task, we used the Turkish dataset to classify data labeled as offensive and non-offensive.

The sample dataset is represented in Table 0.2. Turkish dataset contains 31756 tweets as training set and 3528 tweets as test set. Table 0.3 shows the distribution of labels in this dataset. In the given dataset, volunteers annotated Turkish tweets manually with the following labels.

Offensive (OFF): Tweets that contain inappropriate language, insults, or threats.

Not Offensive (NOT): Tweets that do not contain offensive content.

Table 0.2 Sample Dataset of Tweets

Id	**Tweet**	**Label**
10906	Benden başka herkese iyi geceler	NOT
19321	@USER Burası da fena değil atkafalı	OFF
26169	Ps3 veya ps4 lazım var mı satan?	NOT
30126	elim ayağım titriyor öyle bakma lan vicdansız	OFF

Table 0.3 Distribution of Labels in Tweets

Label	**Data**	
	Train	***Test***
OFF	6129	716
NOT	25627	2812
TOTAL	31756	3528

OFFENSIVE LANGUAGE DETECTION

Initially, we applied pre-processing steps that include converting text to lowercase, removal of punctuation, Twitter-specific “user” token, and symbols like “@” and “#”. After pre-processing, we employ NLTK tokenizer for tokenizing tweets. We ran classical machine learning algorithms to create a baseline and have more detailed information about the dataset. We explored the effects of different fine-tuning methods and pre-trained BERT models for this classification task.

Classical Methods

We experimented with four classical methods: Logistic Regression, Decision Tree, Random Forest, and SVM. We selected Logistic Regression as the first method to work with, Decision Tree and Random Forest to gain intuition about the dataset, SVM due to its wide usage in classification. Models used TF-IDF with the feature set size is equal to 3000. We employed sklearn [79] libraries and set regularization parameter 10 for SVM. We used default values for other features at sklearn 0.24.2 version.

Fine-tuning Methods

In the first part of our study, we focused on different fine-tuning methods. We used fine-tuning method term for the pre-processing method whose effect we wanted to research. This section can be interpreted as feature engineering because we examined the training dataset and searched for ways to increase accuracy. We chose Fixed Prefix Stemmer regarding Turkish language features; we applied Removing Strong Words, Dictionary Definition for Rare Words, Hashtag Segmentation, and Removing Emoji methods by analyzing Twitter data features.

Removing Rare Words

Removing rare words from text is a common pre-processing step. If a rare word happens to be noise, removing it would improve the performance. In addition, rare words do not improve performance and cause computational cost. A recent study [13] shows that removing stop words has little effect on the Turkish classification task. Considering the features of the language used in social media and BERT models, we researched whether results that comply with this study can be obtained. In this step, we experimented with removing words where the occurrence rate is below 0.1.

Fixed Prefix Stemmer

Fixed Prefix Stemmer (FPS) [80] is a stemming technique that takes the first n characters of the specified word. If the word has fewer than n characters, truncation operation would not be performed, and the word is used as it is. Turkish is an agglutinative language; however, the usage of prefixes is uncommon. The idea of FPS is based on the observation that Turkish roots are not affected by changing suffixes. Taking the first five letters of words yields better results than taking the first three or the first seven letters. It is found that stemming and stop word removing have little effect on the classification task. However, the size of the dataset that used in the research was large [13], and we were interested in researching the effect of FPS on a small dataset like the one we used. To observe the effect of FPS, we fine-tuned BERT model by taking the first five letters of each word.

Removing Strong Words

Adding a strong positive word “love” to an offensive text degrades the performance of the models, hence showing the weakness of machine learning models [2]. The word “love” was chosen to attack the hate speeches. We wanted to examine the effect of removing negative words from non-offensive tweets and removing positive words from offensive tweets. We identify negative and positive words by reviewing the frequently used words depending on the tweet type. For example, the word “beyinsiz” was considered negative, while “mutluluk” was considered positive.

Dictionary Definition for Rare Words

Words in natural language follow Zipfian [81] distribution; some words are frequent while most are rare. To learn the embeddings of these words, the model needs to be trained with a large amount of data containing that word. Another option would be treating rare words as OOV words by changing them to “UNK” special token. But this approach causes the loss of the information contained in that specific word. In both cases, the embedding of rare words is poorly optimized. Auxiliary data such as lexical [82] and semantic knowledge [83] provide useful information for learning word embeddings. Producing embeddings from dictionary definitions improves the performance of a downstream task [84]. TDK [85] is a dictionary for the Turkish language. We used their definitions instead of rare words that occurred once in the dataset.

Hashtag Segmentation

Twitter data contains “#” sign that indicates a keyword or topic. Hashtags could be regarded as a group of words that lack space. Changing word boundary by removing whitespace breaks word models [2]. Hashtags could be informative; hence, parsing and obtaining compound words as tokens will increase the data size. For example, in the tweet “#OdtueBaharSenlikleri için herkes kendini paraladı ama unutmasınlarki bu hareketler size taş sopa küfür olarak geri dönecektir..” hashtag refers to a specific event. Therefore, we wanted to examine the effect of segmenting hashtags. We split hashtags into words using the Turkish NLP library [86].

Removing Emoji

Emojis are small pictures used in electronic messages and web pages. They are some of the most common ways to convey emotions and sentiments in social messaging applications. Twitter data contains more emoji than the pre-training dataset of BERT. Some emoji were misused in the training dataset, for example, “Hadi hepimize geçmiş olsun. Benim beyin yine stop etti. Ağlamak üzereyim. @USER 💗💗”. Heart shape does not relate to the meaning of the sentence. Therefore, we wanted to examine the effect of removing emojis.

Emoji2text

Users could complement or emphasize the meaning of the message using emojis. The prevalence of emoji usage in the web [87] has made emoji attractive for various NLP tasks such as sentiment classification [88], human-computer interaction [89], and web mining [90]. Converting emojis to text is a typical pre-processing step in offensive language identification tasks. Translating emoji into words in offensive language [72] leads to a drop in accuracy. The authors of the article argue that some emoji characters are used with different meanings in different contexts. We suggest fine-tuning by replacing emojis with Turkish text equivalents could improve performance. We used an emoji library [91] to obtain the English meanings, then translated the definitions to Turkish. For example, 😀 was converted to Sırıtan Surat (Grinning Face).

Pre-trained BERT Models

In the second part of our study, we focused on BERT models that are pre-trained with the Turkish dataset. For BERT models, we set the learning rate as 2e-5, batch size as 32. We trained the models with 4 epochs.

BERTurk

Turkish BERT model (BERTurk) was pre-trained on 35GB corpus size that contains Oscar Corpus [92], Opus Corpora [93], and Wikipedia dump. The model uses 12 transformer layers. BERTurk models differ in vocabulary size 32k and 128k, both of them have cased and uncased versions.

DistilBERTurk

Large-scale pre-trained language models give state-of-the-art results on NLP tasks but require a lot of data and time to train. Due to the high computational complexity and large storage requirements of these models, they cannot be deployed on low-resource machines. Knowledge distillation [94-96] is a compression technique that trains smaller networks by using larger networks. In the simplest term, the student model is trained to reproduce the behavior of the teacher model by trying to match its weights to the teacher model's weights. The output of the teacher network in the last layer is fed into the student network, and the student corrects the weights according to the errors [96]. The teacher can be a single model, or it can be an ensemble of multiple models (Figure 0.1)

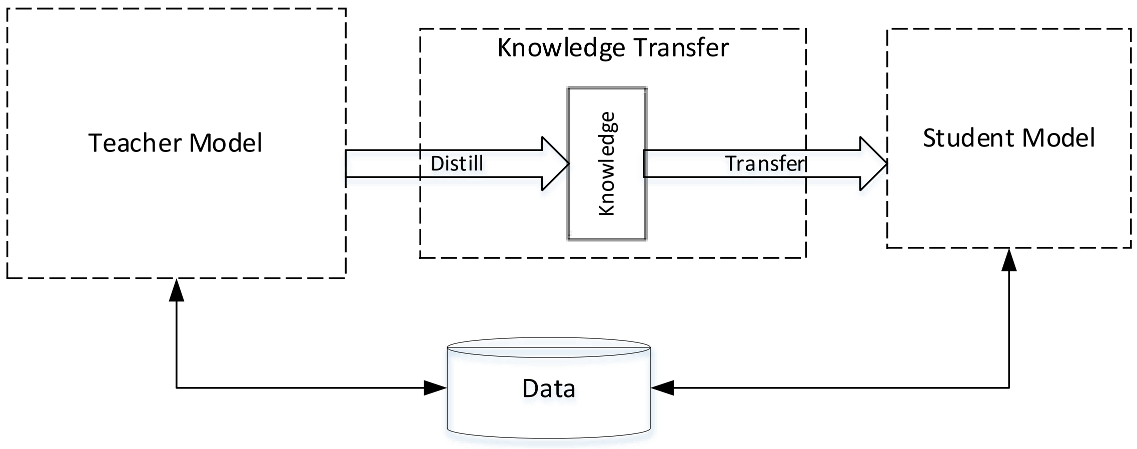

Figure 0.1 The Generic Teacher–Student Framework for Knowledge Distillation, adapted from [96]

DistilBERT [97] was proposed to compress BERT model without much degradation in performance. DistilBERTurk is a version of DistilBERT for the Turkish language. DistilBERTurk was trained on 7GB of the original training data, using the cased version of BERTurk as a teacher model. DistilBERT uses the same pre-training dataset as BERT, it reduces the size of BERT by 40%, and it is 60% faster.

DistilBERT uses soft targets [95] that are probabilities calculated by the softmax function as shown in Eq. 11. z_i is the model score for the i-th class, T is a temperature factor that controls the importance of each soft target.

$$p(z_i, t) = \frac{\exp({}^{z_i}/_{T})}{\sum_j \exp({}^{z_j}/_{T})} \tag{11}$$

The student loss function is defined as cross-entropy cost function (Eq. 12). t_i is a probability estimated by the teacher, and s_i is a probability estimated by the student.

$$L_{ce} = \sum_i t_i * log(s_i) \tag{12}$$

ConvBERTurk

BERT heavily relies on global self-attention blocks. Some of the BERT heads only need to learn local dependencies, which means unnecessary computation costs. ConvBERT [98] was created to remove this redundancy and improve BERT performance on downstream tasks by changing some heads using a natural local operation. Due to the convolution success of extraction of local features, the usage of convolution layers as complementary for self-attention on the pre-training phase was proposed. ConvBERTurk is a version of ConvBERT for the Turkish language, and it was trained on 32k vocabulary cased version for the Turkish language.

ConvBERT enhances novel span-based dynamic convolution. LConv ties all weights along channel dimensions using different kernels at each position. Thus, polysemy words could have different kernel representations. A mixed attention block is a mixture of self-attention and span-based dynamic convolution.

Light-weight convolution is calculated as shown in Eq. 13. X denotes input, W denotes convolution kernel. The output kernel is efficient in modeling local dependencies; however, kernel parameters are fixed. Therefore to capture the diversity of the tokens is problematic.

$$LConv(X, W, i) = \sum_{j=1}^{k} W_j \ . X_{\left(i + j - \frac{k+1}{2}\right)} \qquad (13)$$

Dynamic convolution could learn different kernel parameters for different inputs words (Eq. 14). However, polysemy words have the same parameters, and these parameters are not related to the context of words.

$$DConv(X, W_f, i) = LConv(X, softmax(W_f, X_i)), i) \qquad (14)$$

Span-based dynamic convolution generates different kernel parameters for different spans of words, which means the context of words is also taken into consideration (Eq. 15). ⊙ operator denotes span-based dynamic convolution; K_s is span-based K vector; K, Q, and V are vectors that are obtained by applying linear transformation of input X.

$$SDConv(Q, K_s, V\ ;\ W_f\ , i)\ =\ LConv(V, softmax(W_f\ (Q\ \odot\ K_s)), i) \tag{15}$$

Mixed attention for the word "Car" can be calculated as shown in Eq. 16. Same query, but different keys generate attention maps and convolution kernels. Car(,) denotes concatenation operation.

$$\begin{aligned} &Mixed - Attn(K, Q, K_s, V\ ;\ W_f\ ) \\ &\quad = Car(Self - Attn(Q, K, V\ ), SDConv(Q, K_s, V\ ;\ W_f\ )) \end{aligned} \tag{16}$$

RESULTS AND DISCUSSION

Results of Classical Methods

We shared the performance of classical machine learning methods in Table 0.1. SVM yields the best performance. Since there is a 0.025 difference with SVM and Logistic Regression results, we conclude that the dataset is linearly separable. We experimented with linear, polynomial, radial basis function, sigmoid kernels for SVM and got the best result with linear kernel. The decision tree and random forest results reveal that ensemble methods achieve higher scores than single models. The size of the tweets tagged as non-offensive is around four times the size of offensive tweets (25627 NOT, 6129 OFF). An imbalanced dataset may cause machine learning methods to perform poorly. Resampling the training dataset, using different word representations other than TF-IDF, applying different pre-processing steps, and ensembling multiple models are further research paths.

Table 0.1 Results of Classical Models

Model	**Method**	**F1**
Logistic Regression	Pre-processing	0.622
Decision Tree	Pre-processing	0.629
Random Forest	Pre-processing	0.634
Svm	Pre-processing	0.649

Figure 0.1 shows the performance of classical methods as confusion matrices. Logistic regression made the least error in marking non-offensive tweets as offensive. In addition, it is the most successful classical method in finding non-offensive tweets. Decision tree is the most successful classical method for finding offensive tweets. On the other hand, random forest yields the highest errors in marking offensive tweets as non-offensive.

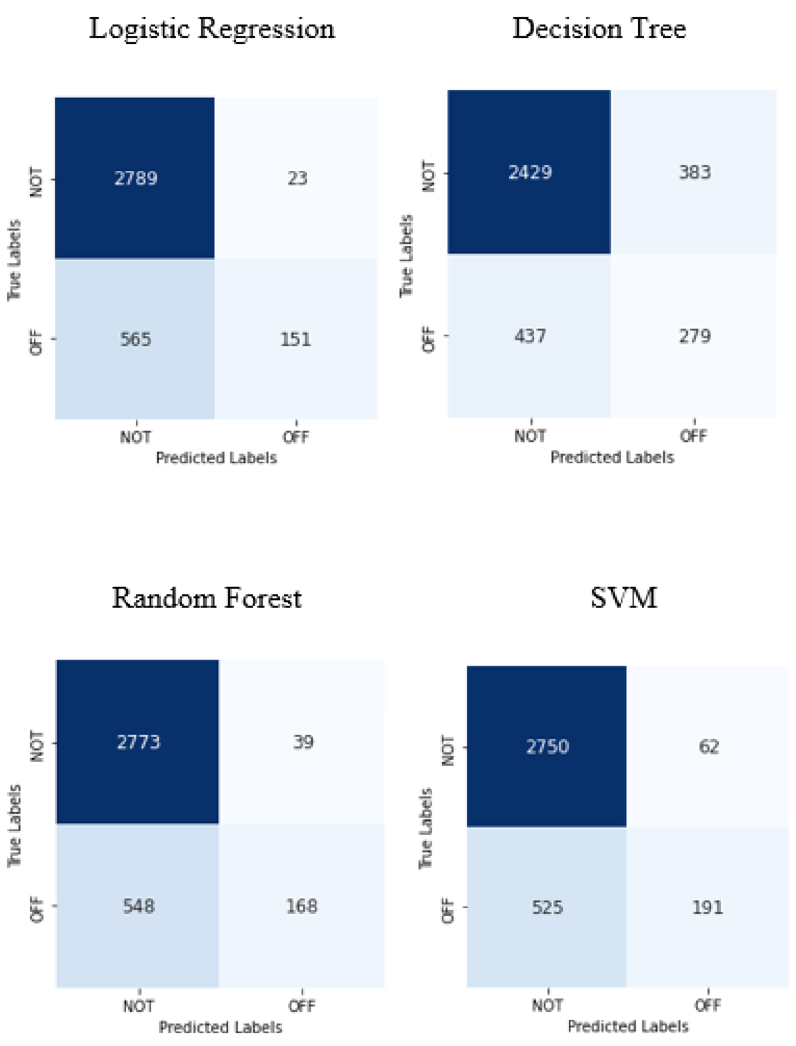

Figure 0.1 Confusion Matrix of Classical Methods

Results of BERT Models

BERT model experimental results are shown in Table 0.2. Experiments point out the importance of the pre-training dataset and the network architecture on which the model is pre-trained. BERTurk (cased, 128k) and ConvBERT obtained the best results. Pre-training BERT models over far more iterations using a larger dataset [13] gains a significant performance boost. mBERT [35] similarly uses a large dataset for pre-training. Performance improvement when adding more information for rare words, substituting emoji with their Turkish equivalents, and hashtag segmentation supports the need for a larger dataset. Models like GPT-3 that were pre-trained using more data would yield even better results. However, studies [99, 100] show that training these models requires substantial energy consumption. Therefore, we regard architectural changes in models as promising for future research. There are BERT models of the same size that yield different results, such as BERTurk (cased, 32k), BERTurk (uncased, 32k), BERTurk (cased, 128k), and BERTurk (uncased, 128k). Uncased versions outperform their cased versions. The possible reason is that BERT tokenizer adds different words to the dictionary for uppercase and lowercase versions of the same word. In addition, on social platforms such as Twitter, users ignore the case sensitivity; using the cased version brings an extra calculation cost.

Fine-tuning methods involving data reduction mostly downgrade the performance except Removing Emoji and Removing Strong Words. Emoji could have been misused by the user, or it is not useful for BERT. Removing Strong Words result is better than BERT base as Grondahl et al. [2] pointed out. The reason for the failure in the FPS model could be that this model breaks the BERT tokenizer. For example, “başkanım” word tokenized as “#başkan” and “#ım” words, while the shortened form “başka” tokenized as “#başka”.

Test data contains swear, misspelled words, and emojis not frequently encountered in newspapers or books. The more frequent a word appears in the pre-training data, the better its representation is learned. A pre-trained model in the target domain [30] increases the model performance. Therefore, a model that is pre-trained in Turkish tweet corpora would improve classification performance.

KUISAIL achieves a 0.814 F1 score for the Turkish language by feeding the last four layers of BERT into a CNN layer. This process happened at the fine-tuning stage. ConvBERT changed some of the attention heads with convolution layers. When comparing KUISAIL and ConvBERT results, we conclude that pre-training model structure is more important than fine-tuning methods.

Character-based models are more resistant to simple evasion attacks based on text transformation than word-based models [2]. Simple evasion attacks are adding letters in words, changing word boundaries by adding or deleting spaces, and adding innocuous words into text. Using character level tokenization instead of wordpiece tokenization could be helpful for classification robustness. CharacterBERT is a character-based BERT implementation. At the time of writing, there was no pre-trained CharacterBERT for the Turkish language.

Table 0.2 Results of BERT Models

Model	**Method**	**F1**
BERTurk (cased, 32k)	Removing Rare Words	0.806
BERTurk (cased, 32k)	Fixed Prefix Stemmer	0.810
BERTurk (cased, 32k)	Pre-processing	0.812
BERTurk (cased, 32k)	Removing Strong Words	0.814
BERTurk (cased, 32k)	Dictionary Definitions for Rare Words	0.814
BERTurk (cased, 32k)	Hashtag Segmentation	0.816
BERTurk (cased, 32k)	Removing Emoji	0.817
BERTurk (cased, 32k)	Emoji2text	0.819
DistilBERTurk	Pre-processing	0.754
BERTurk (uncased, 32k)	Pre-processing	0.818
BERTurk (cased, 128k)	Pre-processing	0.821
BERTurk (uncased, 128k)	Pre-processing	0.823
ConvBERTurk	Pre-processing	0.823

Figure 0.2 shows the confusion matrices of the models that cause the most change in the performance. We compare the performance of the models with the performance of BERTurk (cased, 32k). ConvBERTurk made fewer errors in predicting offensive tweets, and the non-offensive tagging error of offensive tweets has also been reduced. On the other hand, ConvBERT made more errors in predicting non-offensive tweets, and the offensive tagging error of non-offensive tweets has also been increased. BERTurk (uncased, 128k) model outperforms BERTurk (cased, 32k) in any predictions. DistilBERTurk does not exceed BERTurk (cased, 32k) performance on any predictions. Comparing two top methods (BERTurk (uncased, 128k) and ConvBERTurk),

ConvBERTurk's recall score is higher, while BERTurk (uncased, 128k) has a higher precision score.

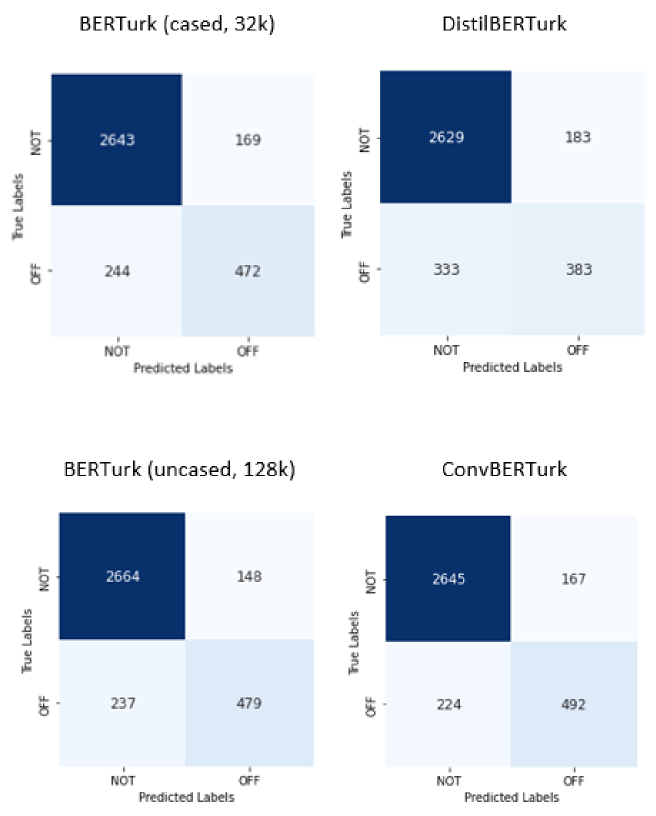

Figure 0.2 Confusion Matrices of BERT Models

Table 5.3 lists our top results and the models that we reviewed. Galileo reached the highest F1 score among the multilingual models. The models it uses are trained in a larger dataset than mBERT, which other multilingual models heavily use, and it ensembles the two models. Other successful models SU-NLP, KUISAIL, and KS@LTH have demonstrated the efficiency of models trained in the language of the downstream task. The training data of the BERT models used by the teams affect the result. For example, while the dictionary size of the BERT model we use is 128k, SU-NLP uses the model with 32k dictionaries.

Table 0.3 Results for Turkish Subtask A

Team	**Score**	**Team**	**Score**
Galileo	0.8258	GruPaTo	0.7790
BERTurk (uncased, 128k)/ConvBERTurk	0.823	INGEOTEC	0.7758
SU-NLP	0.8167	Ferryman	0.7737
KUISAIL	0.8141	ANDES	0.7737
KS@LTH	0.8101	I2C	0.7735
NLPDove	0.7967	LIIR	0.7720
TysonYU	0.7933	LT@Helsinki	0.7719
Rouges	0.7815		

Misclassified Tweets

Table 5.3 shows some of the misclassified tweets and possible reasons. Using innuendo, spelling errors within words, adding spaces within a swear word, using negative words for affection, and using foreign words are possible reasons.

- The polarity of words affects the performance of the model. In short sentences, the effect of the strong positive or strong negative word in the sentence is higher. Using innuendo, using strong positive words in an offensive context, using strong negative words in a non-offensive context, using phrases degrades the performance.
- As the language changes, pre-trained models will need to be re-pre-trained because the context will change. Using context information created after pre-trained models release and foreign words degrades the performance.
- Intentional or accidental typo affects the performance of the model. For example, users may aim to avoid systems that detect swear words by placing spaces between letters in them. Typo cause BERT tokenizer to perform poorly.

Table 0.4 Example of Misclassified Tweets

Tweet	**Label**	**Prediction**	**Cause**
Ekonomi ile top yekün mücadele için bol bol sigara için, her paket de extra 2 TL katkınız olsun...	OFF	NOT	Innuendo
1 hafta sonra gidiyorum sizinde dostluğunuzunda sevginizinde şehrinizinde içine tükürim	OFF	NOT	Typo (tükürim)
ŞU DÜNYA DÜZ DİYENLER Ş İ K T İ R G İ D İ N LÜTFEN	OFF	NOT	Adding spaces
Eski sevgilimi Tinder'da göremem çünkü hornet kullanıyor. Djjdxj keşke şaka olsaydi:(	OFF	NOT	Context information(tinder-hornet)
Ya siz niye böyle kafayı yediniz :(	OFF	NOT	Phrase (kafayı yediniz)
Şeref ekmek bulamazken şerefsiz bulur	NOT	OFF	Negative word (şerefsiz)
Kız haklı dağılın haydi devaam İyikiDoğdun EgeKökenli	NOT	OFF	BERT tokenizer (devaam)
@USER Gardaş dedim bağrıma bastım galleş çıktın püüagg	NOT	OFF	Using negative words for affection (galleş)
Yan komşumuz yine formunda. Küçük çocuğuna; Zaruke xelke mezin dibin akil dibin,yeme mezin dibin din u har dibin 😆	NOT	OFF	Foreign words (Zaruke xelke mezin dibin akil dibin,yeme mezin dibin din u har dibin)

CONCLUSION

In this thesis, we researched the effects of fine-tuning techniques and different BERT models on the Turkish offensive language classification task. We developed our fine-tuning methods by analyzing the dataset, and we experimented with BERT models that pre-trained on Turkish corpora.

Besides BERT models, we also implemented some of the classic methods, and as we expected, we achieved lower F1 scores than all BERT models. SVM yields the highest score among classical methods. We suggested possible improvement paths such as; resampling the training dataset, using different word representations other than TF-IDF, applying different pre-processing steps, and ensembling multiple models.

In general, fine-tuning methods that reduce the dataset downgrade performance, while methods that increase the dataset provide a slight improvement. Although the fine-tuning techniques that aimed to take advantage of the features of Twitter data affected the success of the model, they were not as effective as the size or the architecture of BERT models.

Experimental results show the importance of the pre-trained model. Pre-training BERT models over far more iterations using a larger dataset and improving the deficiencies in BERT architecture yields comparable performance to best systems on the Turkish language dataset. However, we are confident that modifying the model architectures will be more appropriate for future research since pre-training larger models will demand higher energy consumption. A pre-trained model in the target domain increases the model performance. Therefore, a model that is pre-trained in Turkish tweet corpora would improve classification performance.

We analyzed the misclassified data and listed probable reasons, such as; using innuendo, using rare words that do not appear in the pre-training dataset, and typos. We concluded that the user would also desire to deceive offensive language detection systems. For example, the user can change a letter in a word and break the model's word tokenization process. In such cases, we expect that using character-based models such as CharacterBERT will achieve higher accuracy.

REFERENCES

[1] A. Schmidt, M. Wiegand, A survey on hate speech detection using natural language processing, Proceedings of the fifth international workshop on natural language processing for social media, 2017, pp. 1-10.

[2] T. Grondahl, L. Pajola, M. Juuti, M. Conti, N. Asokan, All You Need is "Love": Evading Hate Speech Detection, Aisec'18: Proceedings of the 11th Acm Workshop on Artificial Intelligence and Security, (2018) 2-12.

[3] T. Davidson, D. Warmsley, M. Macy, I. Weber, Automated hate speech detection and the problem of offensive language, Proceedings of the International AAAI Conference on Web and Social Media, 2017.

[4] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, Y. Chang, Abusive Language Detection in Online User Content, Proceedings of the 25th International Conference on World Wide Web (Www'16), (2016) 145-153.

[5] H. Hosseinmardi, S.A. Mattson, R.I. Rafiq, R. Han, Q. Lv, S. Mishra, Analyzing labeled cyberbullying incidents on the instagram social network, International conference on social informatics, Springer, 2015, pp. 49-66.

[6] P. Fortuna, S. Nunes, A survey on automatic detection of hate speech in text, ACM Computing Surveys (CSUR), 51 (2018) 1-30.

[7] F. Poletto, V. Basile, M. Sanguinetti, C. Bosco, V. Patti, Resources and benchmark corpora for hate speech detection: a systematic review, Language Resources and Evaluation, 55 (2021) 477-523.

[8] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval), arXiv preprint arXiv:1903.08983, (2019).

[9] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, L. Derczynski, Z. Pitenis, Ç. Çöltekin, SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020), arXiv preprint arXiv:2006.07235, (2020).

[10] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep

bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805, (2018).

[11] Ç. Çöltekin, A corpus of Turkish offensive language on social media, Proceedings of the 12th Language Resources and Evaluation Conference, 2020, pp. 6174-6184.

[12] S. Schweter, BERTurk - BERT models for Turkish [online] Website https://doi.org/10.5281/zenodo.3770924 [accessed: 10.2021].

[13] D. Torunoğlu, E. Çakirman, M.C. Ganiz, S. Akyokuş, M.Z. Gürbüz, Analysis of preprocessing methods on classification of Turkish texts, 2011 International Symposium on Innovations in Intelligent Systems and Applications, IEEE, 2011, pp. 112-117.

[14] M. Toman, R. Tesar, K. Jezek, Influence of word normalization on text classification, Proceedings of InSciT, 4 (2006) 354-358.

[15] M. Braschler, B. Ripplinger, How effective is stemming and decompounding for German text retrieval?, Information Retrieval, 7 (2004) 291-316.

[16] C.-Y.J. Peng, K.L. Lee, G.M. Ingersoll, An introduction to logistic regression analysis and reporting, The journal of educational research, 96 (2002) 3-14.

[17] F.-J. Yang, An extended idea about decision trees, 2019 International Conference on Computational Science and Computational Intelligence (CSCI), IEEE, 2019, pp. 349-354.

[18] L. Breiman, Random forests, Machine learning, 45 (2001) 5-32.

[19] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, Proceedings of the fifth annual workshop on Computational learning theory, 1992, pp. 144-152.

[20] X.P. Qiu, T.X. Sun, Y.G. Xu, Y.F. Shao, N. Dai, X.J. Huang, Pre-trained models for natural language processing: A survey, Sci China Technol Sc, 63 (2020) 1872-1897.

[21] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science, 313 (2006) 504-507.

[22] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing

systems, 25 (2012) 1097-1105.

[23] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, Advances in neural information processing systems, 2013, pp. 3111-3119.

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

[25] Y. Kim, Y. Jernite, D. Sontag, A.M. Rush, Character-Aware Neural Language Models, Thirtieth Aaai Conference on Artificial Intelligence, (2016) 2741-2749.

[26] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics, 5 (2017) 135-146.

[27] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, arXiv preprint arXiv:1802.05365, (2018).

[28] J. Howard, S. Ruder, Universal language model fine-tuning for text classification, arXiv preprint arXiv:1801.06146, (2018).

[29] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training, (2018).

[30] Z.H. Dai, Z.L. Yang, Y.M. Yang, J. Carbonell, Q.V. Le, R. Salakhutdinov, Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context, 57th Annual Meeting of the Association for Computational Linguistics (Acl 2019), (2019) 2978-2988.

[31] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692, (2019).

[32] Z.L. Yang, Z.H. Dai, Y.M. Yang, J. Carbonell, R. Salakhutdinov, Q.V. Le, XLNet: Generalized Autoregressive Pretraining for Language Understanding, Adv Neur In, 32 (2019).

[33] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite

bert for self-supervised learning of language representations, arXiv preprint arXiv:1909.11942, (2019).

[34] G. Lample, A. Conneau, Cross-lingual language model pretraining, arXiv preprint arXiv:1901.07291, (2019).

[35] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, Unsupervised cross-lingual representation learning at scale, arXiv preprint arXiv:1911.02116, (2019).

[36] Q. Yang, Y. Zhang, W. Dai, S.J. Pan, Transfer learning, Cambridge University Press2020.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention Is All You Need, Advances in Neural Information Processing Systems 30 (Nips 2017), 30 (2017).

[38] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473, (2014).

[39] Tensor2Tensor [online] Website https://sozluk.gov.tr [accessed: 10.2021].

[40] W.L. Taylor, “Cloze procedure”: A new tool for measuring readability, Journalism quarterly, 30 (1953) 415-433.

[41] M. Schuster, K. Nakajima, Japanese and korean voice search, 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2012, pp. 5149-5152.

[42] D. Fišer, R. Huang, V. Prabhakaran, R. Voigt, Z. Waseem, J. Wernimont, Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), 2018.

[43] R. Kumar, A.K. Ojha, S. Malmasi, M. Zampieri, Benchmarking aggression identification in social media, Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), 2018, pp. 1-11.

[44] R. Kumar, A.K. Ojha, S. Malmasi, M. Zampieri, Evaluating aggression identification in social media, Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying, 2020, pp. 1-5.

[45] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, Predicting

the type and target of offensive posts in social media, arXiv preprint arXiv:1902.09666, (2019).

[46] P. Liu, W. Li, L. Zou, NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers, Proceedings of the 13th international workshop on semantic evaluation, 2019, pp. 87-91.

[47] A. Nikolov, V. Radivchev, Nikolov-radivchev at semeval-2019 task 6: Offensive tweet classification with bert and ensembles, Proceedings of the 13th International Workshop on Semantic Evaluation, 2019, pp. 691-695.

[48] A. Seganti, H. Sobol, I. Orlova, H. Kim, J. Staniszewski, T. Krumholc, K. Koziel, NLPR@ SRPOL at SemEval-2019 Task 6 and Task 5: Linguistically enhanced deep learning offensive sentence classifier, arXiv preprint arXiv:1904.05152, (2019).

[49] J. Zhu, Z. Tian, S. Kübler, UM-IU@ LING at SemEval-2019 task 6: Identifying offensive tweets using BERT and SVMs, arXiv preprint arXiv:1904.03450, (2019).

[50] A. Pelicon, M. Martinc, P.K. Novak, Embeddia at semeval-2019 task 6: Detecting hate with neural network and transfer learning approaches, Proceedings of the 13th International Workshop on Semantic Evaluation, 2019, pp. 604-610.

[51] K. Socha, KS@ LTH at SemEval-2020 Task 12: Fine-tuning multi-and monolingual transformer models for offensive language detection, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 2045-2053.

[52] M. Pàmies, E. Öhman, K. Kajava, J. Tiedemann, LT@ Helsinki at SemEval-2020 Task 12: Multilingual or language-specific BERT?, arXiv preprint arXiv:2008.00805, (2020).

[53] H. Alami, S.O. El Alaoui, A. Benlahbib, N. En-nahnahi, LISAC FSDM-USMBA Team at SemEval-2020 Task 12: Overcoming AraBERT's pretrain-finetune discrepancy for Arabic offensive language identification, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 2080-2085.

[54] A. Ozdemir, R. Yeniterzi, SU-NLP at SemEval-2020 Task 12: Offensive language identification in Turkish tweets, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 2171-2176.

[55] A. Safaya, M. Abdullatif, D. Yuret, Kuisail at semeval-2020 task 12: Bert-cnn for offensive speech identification in social media, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 2054-2059.

[56] P.C. Chen, H.-H. Huang, H.-H. Chen, NTU_NLP at SemEval-2020 Task 12: Identifying Offensive Tweets Using Hierarchical Multi-Task Learning Approach, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 2105-2110.

[57] J.M. Pérez, A. Arango, F. Luque, ANDES at SemEval-2020 Task 12: A jointly-trained BERT multilingual model for offensive language detection, arXiv preprint arXiv:2008.06408, (2020).

[58] S. Wang, J. Liu, X. Ouyang, Y. Sun, Galileo at SemEval-2020 task 12: Multi-lingual learning for offensive language identification using pre-trained language models, arXiv preprint arXiv:2010.03542, (2020).

[59] T. Dadu, K. Pant, Team Rouges at SemEval-2020 Task 12: Cross-lingual inductive transfer to detect offensive language, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 2183-2189.

[60] H. Ahn, J. Sun, C.Y. Park, J. Seo, NLPDove at SemEval-2020 task 12: Improving offensive language detection with cross-lingual transfer, arXiv preprint arXiv:2008.01354, (2020).

[61] D. Colla, T. Caselli, V. Basile, J. Mitrović, M. Granitzer, Grupato at semeval-2020 task 12: Retraining mbert on social media and fine-tuned offensive language models, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 1546-1554.

[62] V.P. Álvarez, J.M. Vázquez, J.M.L. Betanzos, J.L.A. Fernández, I2C at SemEval-2020 Task 12: Simple but Effective Approaches to Offensive Speech Detection in Twitter, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 1968-1977.

[63] E. Ghadery, M.-F. Moens, Liir at semeval-2020 task 12: A cross-lingual augmentation approach for multilingual offensive language identification, arXiv preprint arXiv:2005.03695, (2020).

[64] S. Hassan, Y. Samih, H. Mubarak, A. Abdelali, ALT at SemEval-2020 task 12:

Arabic and English offensive language identification in social media, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 1891-1897.

[65] G. Wiedemann, S.M. Yimam, C. Biemann, UHH-LT at SemEval-2020 Task 12: Fine-Tuning of Pre-Trained Transformer Networks for Offensive Language Detection, SEMEVAL, 2020.

[66] S. Sotudeh, T. Xiang, H.-R. Yao, S. MacAvaney, E. Yang, N. Goharian, O. Frieder, Guir at semeval-2020 task 12: Domain-tuned contextualized models for offensive language detection, arXiv preprint arXiv:2007.14477, (2020).

[67] W. Chen, P. Wang, J. Li, Y. Zheng, Y. Wang, Y. Zhang, Ferryman at SemEval-2020 Task 12: BERT-Based Model with Advanced Improvement Methods for Multilingual Offensive Language Identification, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 1947-1952.

[68] A. Mosquera, amsqr at SemEval-2020 Task 12: Offensive language detection using neural networks and anti-adversarial features, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 1898-1905.

[69] D. Mahata, H. Zhang, K. Uppal, Y. Kumar, R. Shah, S. Shahid, L. Mehnaz, S. Anand, MIDAS at SemEval-2019 task 6: Identifying offensive posts and targeted offense from twitter, Proceedings of the 13th International Workshop on Semantic Evaluation, 2019, pp. 683-690.

[70] M. Uzan, Y. HaCohen-Kerner, JCT at SemEval-2020 Task 12: Offensive language detection in tweets using preprocessing methods, character and word n-grams, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 2017-2022.

[71] S. Miranda-Jiménez, E.S. Tellez, M. Graff, D. Moctezuma, INGEOTEC at SemEval-2020 Task 12: Multilingual classification of offensive text, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 1992-1997.

[72] Z. Wu, H. Zheng, J. Wang, W. Su, J. Fong, Bnu-hkbu uic nlp team 2 at semeval-2019 task 6: Detecting offensive language using bert model, Proceedings of the 13th International Workshop on Semantic Evaluation, 2019, pp. 551-555.

[73] B. Wang, X. Zhou, X. Zhang, YNUWB at SemEval-2019 Task 6: K-max pooling CNN with average meta-embedding for identifying offensive language,

Proceedings of the 13th International Workshop on Semantic Evaluation, 2019, pp. 818-822.

[74] S.W.C. Wang, Offensive language classification in social media: using deep learning, 2020.

[75] W. Dai, T. Yu, Z. Liu, P. Fung, Kungfupanda at semeval-2020 task 12: Bert-based multi-task learning for offensive language detection, arXiv preprint arXiv:2004.13432, (2020).

[76] M. Ibrahim, M. Torki, N.M. El-Makky, AlexU-BackTranslation-TL at SemEval-2020 Task 12: Improving offensive language detection using data augmentation and transfer learning, Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2020, pp. 1881-1890.

[77] A.I. Alharbi, M. Lee, Combining character and word embeddings for the detection of offensive language in Arabic, Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, 2020, pp. 91-96.

[78] H.E. Boukkouri, O. Ferret, T. Lavergne, H. Noji, P. Zweigenbaum, J. Tsujii, CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters, arXiv preprint arXiv:2010.10392, (2020).

[79] scikit-learn: machine learning in Python [online] Website https://scikit-learn.org/ [accessed: 10.2021].

[80] F. Can, S. Kocberber, E. Balcik, C. Kaynak, H.C. Ocalan, O.M. Vursavas, Information retrieval on Turkish texts, Journal of the American Society for Information Science and Technology, 59 (2008) 407-421.

[81] G.K. Zipf, Human behavior and the principle of least effort: An introduction to human ecology, Ravenio Books2016.

[82] C. Xu, Y. Bai, J. Bian, B. Gao, G. Wang, X. Liu, T.-Y. Liu, Rc-net: A general framework for incorporating knowledge into word representations, Proceedings of the 23rd ACM international conference on conference on information and knowledge management, 2014, pp. 1219-1228.

[83] M. Faruqui, J. Dodge, S.K. Jauhar, C. Dyer, E. Hovy, N.A. Smith, Retrofitting word vectors to semantic lexicons, arXiv preprint arXiv:1411.4166, (2014).

[84] D. Bahdanau, T. Bosc, S. Jastrzębski, E. Grefenstette, P. Vincent, Y. Bengio, Learning to compute word embeddings on the fly, arXiv preprint arXiv:1706.00286, (2017).

[85] Türk Dil Kurumu | Sözlük [online] Website https://sozluk.gov.tr [accessed: 10.2021].

[86] M. Çetinkaya, GitHub - MeteHanC/turkishnlp: Very early version of the TurkishNLP. [online] Website https://github.com/MeteHanC/turkishnlp [accessed: 10.2021].

[87] M. Li, E. Chng, A.Y.L. Chong, S. See, An empirical analysis of emoji usage on Twitter, Industrial Management & Data Systems, (2019).

[88] Z. Chen, S. Shen, Z. Hu, X. Lu, Q. Mei, X. Liu, Emoji-powered representation learning for cross-lingual sentiment classification, The World Wide Web Conference, 2019, pp. 251-262.

[89] A. Beattie, A.P. Edwards, C. Edwards, A bot and a smile: Interpersonal impressions of chatbots and humans using emoji in computer-mediated communication, Communication Studies, 71 (2020) 409-427.

[90] H.J. Miller, J. Thebault-Spieker, S. Chang, I. Johnson, L. Terveen, B. Hecht, "Blissfully Happy" or "Ready toFight": Varying Interpretations of Emoji, Tenth international AAAI conference on Web and social media, 2016.

[91] K.W. Taehoon Kim, GitHub - carpedm20/emoji: emoji terminal output for Python [online] Website https://github.com/carpedm20/emoji [accessed: 10.2021].

[92] B. Sagot, OSCAR | Open Super-large Crawled Aggregated coRpus. [online] Website https://oscar-corpus.com [accessed: 10.2021].

[93] OPUS - an open source parallel corpus [online] Website https://opus.nlpl.eu [accessed: 10.2021].

[94] C. Buciluǎ, R. Caruana, A. Niculescu-Mizil, Model compression, Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006, pp. 535-541.

[95] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531, (2015).

[96] J. Gou, B. Yu, S.J. Maybank, D. Tao, Knowledge distillation: A survey, International Journal of Computer Vision, 129 (2021) 1789-1819.

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

[98] Z. Jiang, W. Yu, D. Zhou, Y. Chen, J. Feng, S. Yan, Convbert: Improving bert with span-based dynamic convolution, arXiv preprint arXiv:2008.02496, (2020).

[99] E. Strubell, A. Ganesh, A. McCallum, Energy and policy considerations for deep learning in NLP, arXiv preprint arXiv:1906.02243, (2019).

[100] R. Schwartz, J. Dodge, N.A. Smith, O. Etzioni, Green ai, Communications of the ACM, 63 (2020) 54-63.

APPENDICES

APPENDIX 1 – Proceeding that has been accepted for the publication

Özberk, A., & Çiçekli, İ. (2021, September). Offensive Language Detection in Turkish Tweets with Bert Models. In *2021 6th International Conference on Computer Science and Engineering (UBMK)* (pp. 517-521). IEEE.