A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media

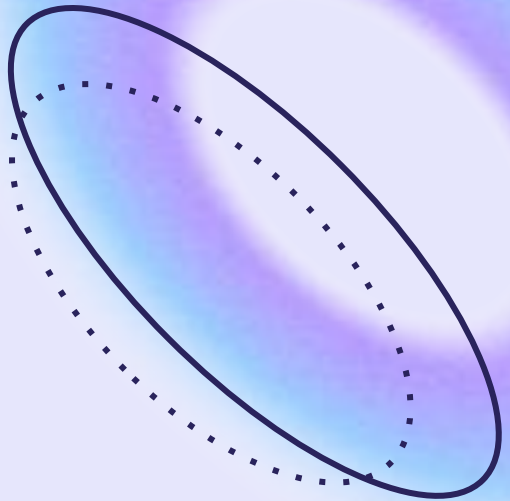
Andrea Adrignola
Amri Myftija
Faezeh Kazemihatami

Table of contents

01.

Abstract

In this section, we present an abstract of what we have done in this project.



Introduction

The detection of hate speech in social medias like twitter is an important task. The spread of hate in social medias platform can have harmful effects on the society using the social medias like twitter.

Abstract

The task to do is to detect online hate speech without miss-classifying non-hateful speech in the process.

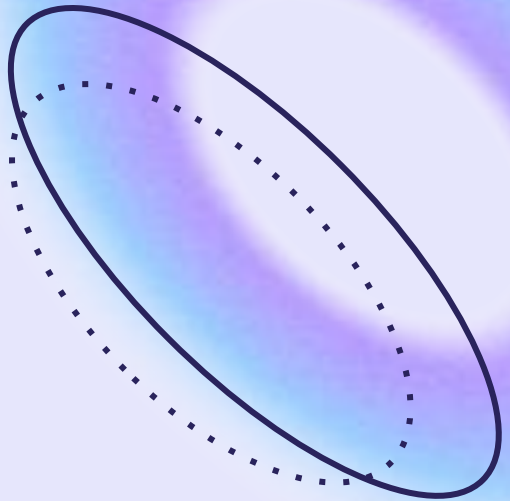
For this project, We have followed Mozafari et al. approach and methodology.

Following the paper's approach, we use a transfer learning methodology starting from a pre-trained BERT model, which will be discussed in detail in the following slides.

02.

Problem Statement

In this section, we explain the problem and its details.



Problem Statement

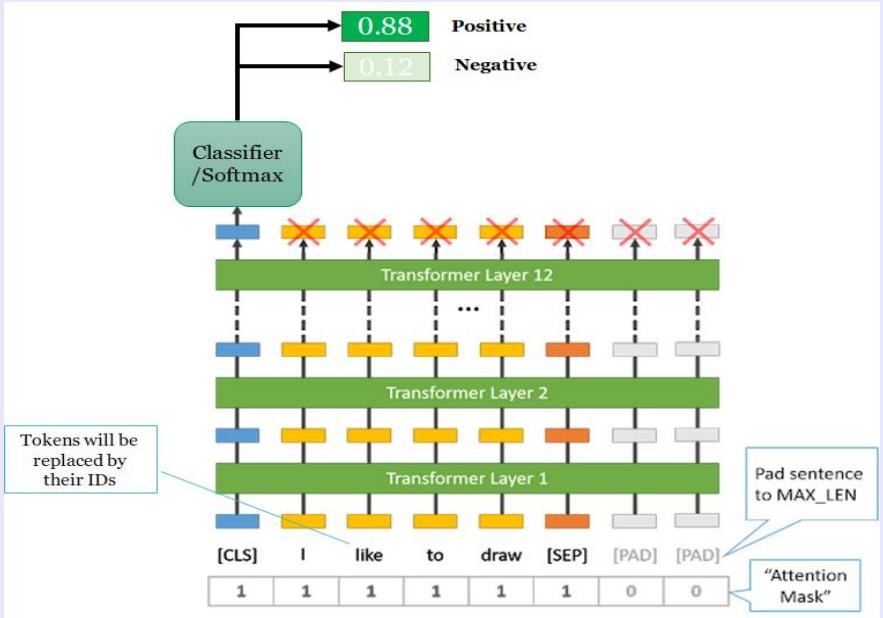
Our goal is to classify social media posts into hateful and non-hateful categories for moderation purposes. As there is a huge amount of posts being generated, we want our process to be automated.

We describe our problem as follows:

- expected input:d, short-length text document, such as atweet
- addressed task: classify d into the specified categories,vaguely described as:{hateful, non_hateful}. A high precision and recall are desired.
- expected output:label(d)

Problem Statement

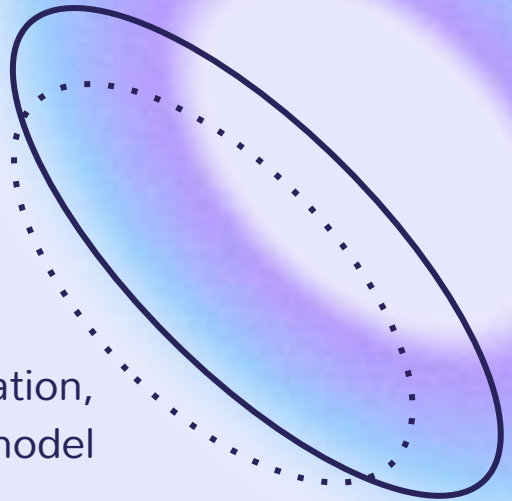
To analyze the ability of the BERT transformer model on the identification of hate speech, we describe the mechanism used in the pre-trained BERT model.



03.

Methodology

In this section, we discuss the methods used in the project.
The pipeline used consists of text pre-processing,tokenization, followed by training and evaluation of the different model configurations.



Pre-processing

- Case normalization
- Removal of special characters
- Handling non-texual data

Pre-processing

- Converting all tweets to lower case
- Removing mentions of users
- Removing embedded URLs in tweets' content
- Removing common emoticons, because in this study we do not consider emotions in our analysis
- Identifying elongated words and converting them into short and standard format
- Removing hashtag signs (\#) and replacing the hashtag texts by their textual counterparts
- Removing all punctuation marks, unknown uni-codes and extra delimiting characters
- Keeping all stop words, because our model trains the sequence of words in a text directly
- Remove tweets shorter than two words
- Finally, a [CLS] token is added at the start of each tweet (this is required for performing classification with BERT)

Elongated words

1. Search for non-standard expressions
2. If they are found, remove the repetitions and check if the reduced version is in the vocabulary
3. If the word is found in the vocabulary, then keep it

Hashtags

1. Search for “<hashtags>” tags
2. Infer spaces in the word right after the tag
3. Keep the spaced version

Tokenization

Divide sentences into tokens using pre-trained BERT, max lenght 64 because usually tweets aren’t very long

Model Configurations

BERT

The standard use of BERT for classification

BERT→NL network

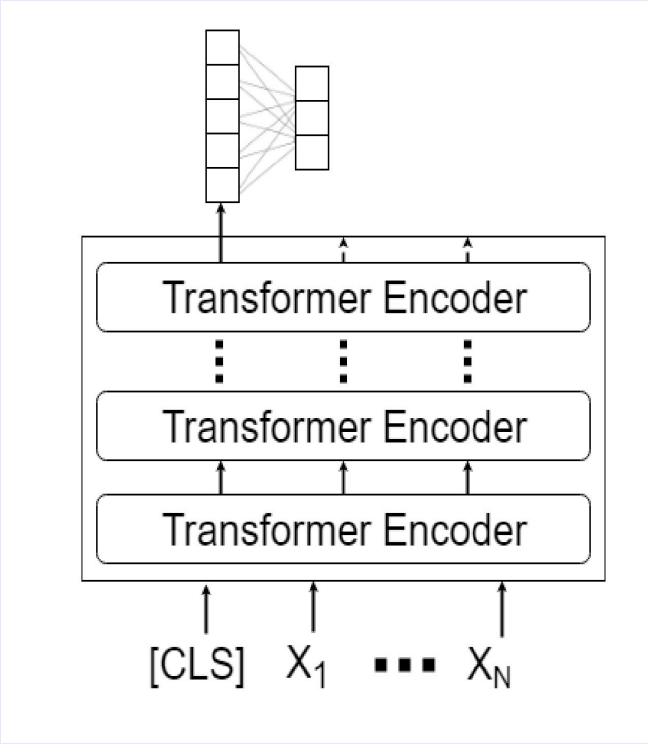
The first architecture is upgraded and an architecture with a more robust classier is provided.

BERT→BiLSTM

in this architecture all outputs of the latest transformer encoder are used.

BERT + CNN

The outputs of all transformer encoders are used instead of using the output of the latest transformer encoder.

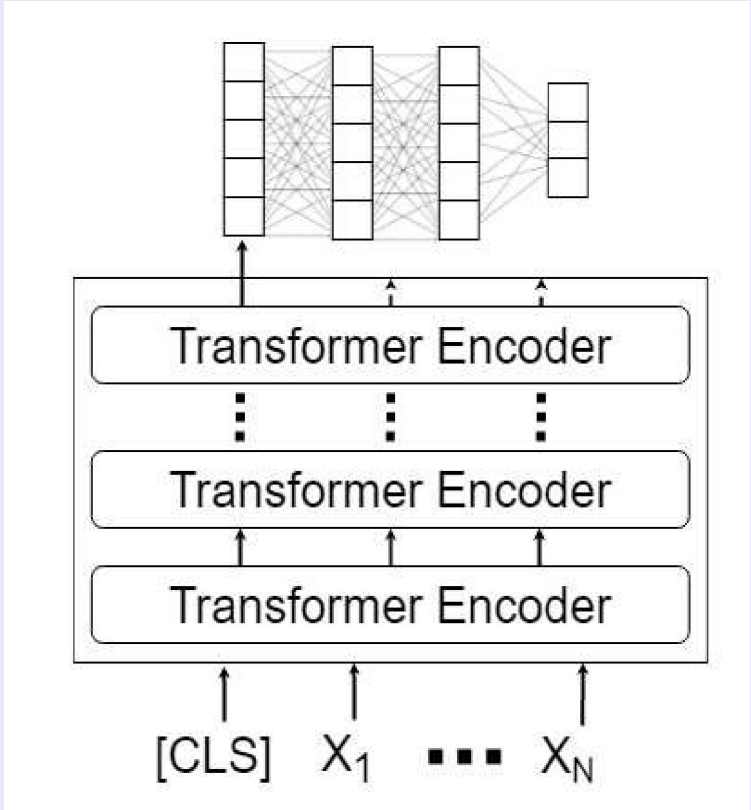


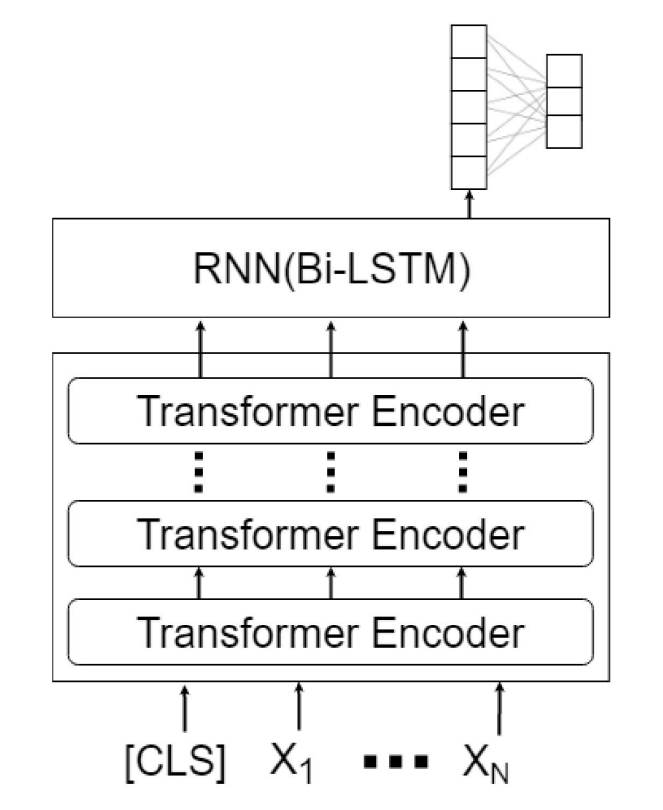
Bert-Base

This is the standard use of BERT for classification. The [CLS] output of the 12th (last) transformer encoder, a vector of size 768, is passed through a softmax function in order to perform classification.

BERT→NL network

The last encoder's [CLS] output is passed through a two-layered, fully-connected network with hidden layers of size 768.
It uses the Leaky ReLU activation function and p_dropout=0,1
Finally, a softmax is applied for classification.





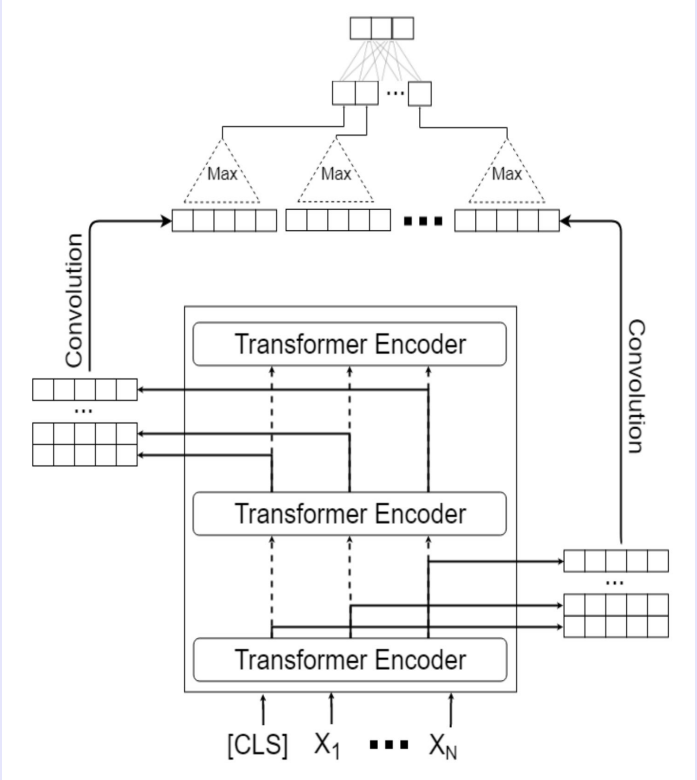
BERT→BiLSTM

Alll of the latest encoder's outputs are fed into a BiLSTM of hidden size 768.

The BiLSTM output is fed to a softmax for classification

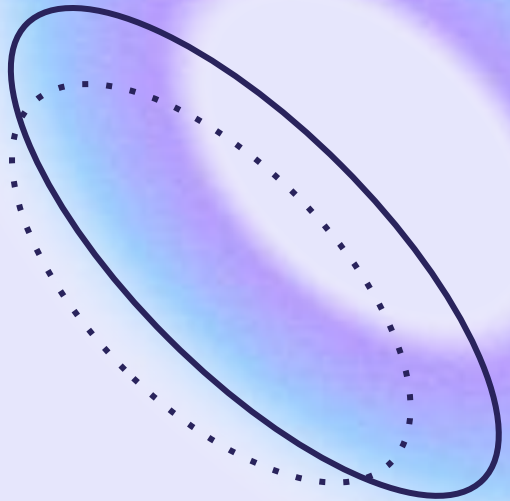
BERT + CNN

The outputs of all 12 hidden states are passed to a CNN with the following characteristics: window size 3, max pooling, p_dropout=0,1
Finally, the output of the CNN is passed to a softmax for classification



04.

Experiments



Data Description

Davidson dataset structure:

<index | annotater_count | hate_count | offensive_count | neither_count | class | tweet >

Class representation:

Hate: 6%

Offensive: 77%

Neither: 17%

	Hate	Offensive	Neither
Training	1144	15352	3331
Validation	143	1919	416
Test	143	1919	416
Total	1430	19190	4163

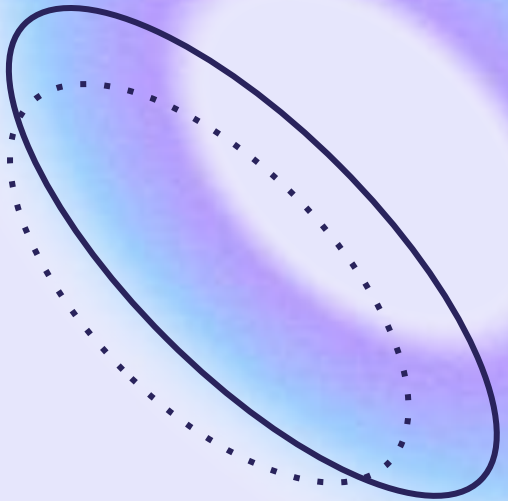
Experimental Design

Hardware	Google Colab
Software/Libraries	Pandas, Numpy, Pytorch, Transformers, Sklearn
Validation method	Test/Train split - Holdout; 10% of the data used for testing
Performance Metrics	Weighted F1-score

05.

Results

In this section, we discuss the training times and the evaluations for the four configurations.



Results

	Weighted F1	Training Time
BERT	**0.85**	5.15
BERT -> NN	**0.86**	5.32
BERT -> BiLSTM	~~**0.67**~~ **0.88**	~~5.58~~ 10.72
BERT + CNN	**0.89**	21.96

Thanks for your attention