Approaches to Detecting Online Abuse Using Machine Learning

M.Srividya

Assistant Professor

Department of Information Technology

Matrusri Engineering College

email:m.srividyareddy@matrusri.edu.in

Abstract: HateSpeech (HS) is a complex phenomenon on social media platforms like Twitter, garnering significant attention in Natural Language Processing (NLP) research. HS Spreaders, often referred to as "haters," actively propagate hate speech through these platforms. This study focuses on the identification of such individuals.

Our approach involves two key components. First, we employ a class-dependent Lower Dimensionality Statistical Embedding (LDSE) representation, which is subsequently input to a linear Support Vector Machine (SVM) classifier. This step helps identify haters based on general commonalities. Second, we capture the stylistic features of individuals by employing extractive summarization of their tweets in conjunction with RoBERTa embedding, followed by classification using another linear SVM classifier.

1.Introduction

Social media platforms such as Twitter, Facebook, YouTube, and Instagram are widely used by millions of users for the public sharing of user-generated content. These platforms serve as vibrant spaces for user engagement and discussions. Unfortunately, participating in online interactions exposes individuals to an ongoing risk of being targeted or harassed, often subjected to abusive language and expressions of hatred, including racism or sexism. Such behaviour can have adverse consequences not only for the individual but also for the broader community.

The challenge we face is the formulation of effective policies to identify and respond appropriately to instances of harassment. This challenge is compounded by the inherent complexity of comprehensively studying these phenomena on a large scale. One particularly troubling manifestation is hate speech, commonly defined as any communication that denigrates an individual or a group based on specific characteristics such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other defining attributes.

In response to these formidable challenges, our research centers on a critical endeavour known as author profiling. Our specific goal is the identification of potential instigators of hate speech, often referred to as "haters," on the Twitter platform. Our aim is to initiate actions aimed at curtailing the spread of hate speech among online users. It is noteworthy that this undertaking embraces a multilingual perspective, encompassing both the English and Spanish languages.

In their research, the authors reached a noteworthy conclusion regarding traditional word embedding models, specifically GloVe[3] and Word2Vec[4], when applied to the challenge of toxic comment classification. These conventional models encounter difficulties in handling the out-of-vocabulary problem, particularly when faced with words not included in their pre-trained vocabulary. However, the authors observed that word embeddings like FastText[5], which encompass sub word information, exhibit greater suitability for this task. FastText's capacity to manage unknown words renders previous assertions about the inferiority of word embeddings, in contrast to word n-grams, outdated.

Additionally, the introduction of contextualized word embeddings, exemplified by BERT, has ushered in a significant advancement in the realm of natural language processing. These contextual embeddings effectively address several limitations inherent in traditional word embeddings.

In the specific context of the SemEval-2021 Toxic Span Detection task [7], researchers opted for a combination of diverse word embeddings, encompassing GloVe, GPT-2[8], and RoBERTa[9]. This amalgamation of embeddings was strategically employed to generate fortified representations that adeptly surmount the out-of-vocabulary challenge. The researchers posited that words of a derogatory nature might not have been included

during the training of word embeddings like GloVe or Word2Vec. However, the utility of these embeddings becomes apparent when they are integrated with other representations. The concatenation of these embeddings with language model (LM) models serves to convey essential awareness signals to the overarching model.

2.Proposed Approach

Our proposed approach aims to predict whether the user is keen to spread hate speech or not. We utilized two distinct representations: one based on contextualized embeddings from RoBERTa and another employing lower-dimensional statistical embeddings generated from character-level data, referred to as Char-LDSE."

2.1Contextualized Representation

Contextualized representation is a technique used in natural language processing (NLP) to create word or phrase embeddings that consider the context in which they appear within a text. This approach differs from traditional static word embeddings, which assign fixed vector representations to words without considering context.

Instead, contextualized embeddings adapt their vectors based on the surrounding words in a sentence or document. This dynamic adjustment allows them to capture the meaning of a word or phrase in various contexts, making them highly valuable for a wide range of NLP tasks.

One of the most prominent examples of contextualized embeddings is BERT (Bidirectional Encoder Representations from Transformers). BERT models are pre-trained on extensive text data and are designed to generate embeddings that take into account both the left and right context of each word in a sentence. This bidirectional context consideration enables BERT embeddings to capture subtle nuances in language and meaning that traditional word embeddings cannot.

Contextualized representations have found widespread applications across various NLP tasks, including text classification, sentiment analysis, named entity recognition, machine translation, and more. Researchers and practitioners often fine-tune pre-trained contextualized models for specific tasks, leveraging their unique ability to understand context and semantics within natural language text.

Text Summarization: To distil the user preferences from the tweets and reduce the volume of text, we employed extractive text summarization. This process generated concise summaries while preserving essential information.

Word Embeddings: We leveraged RoBERTa embeddings to create contextualized representations of user preferences. RoBERTa embeddings are known for their ability to capture contextual information within text effectively.

User Representations: As a result, each user's preferences were represented as vectors of size 768. These vectors encapsulated the essence of the summarized user preferences through word embeddings.

Classification Module: Finally, these 768-dimensional representations served as input for our classification module. This module was responsible for tasks such as sentiment analysis, topic classification, or other forms of user preference categorization, utilizing the contextualized representations.

Extractive Text Summarization:

The process of summarizing user tweets using Gensim's TextRank algorithm involves several steps:

1.Pre-processing the given tweets: Gensim has built-in preprocessing for text data.

2.Creating a graph with tweets as vertices: Each tweet is represented as a node in the graph.

3.Adding edges to represent similarity: Edges are added between tweets to denote their similarity.

4.Running the PageRank algorithm: The PageRank algorithm is applied to this weighted graph.

5.Selecting the highest-scoring vertices: Tweets with the highest PageRank scores are selected.

6.Determining the number of vertices based on a ratio or word count: The number of tweets to be included in the summary is decided based on a specified ratio or word count.

So, the key steps in this summarization process involve preprocessing, graph creation, similarity calculation, PageRank computation, vertex selection, and determining the number of vertices for the final summary.

Preprocessing: After generating the summaries, we conducted preprocessing on this condensed text. This step encompassed tasks such as removing stop words, tokenization, and formatting the text appropriately for further analysis.

In the preprocessing step for summarizing user tweets using Gensim's TextRank algorithm, the following actions are taken on each tweet:

Removal of URLs: Any URLs present in the tweets are removed.

Handling of Hashtags: Hashtags are replaced with the tag "#HASHTAG#."

Handling of Mentions: Mentions are replaced with the tag "#USER#."

Removal of Reserved Words: Reserved words like "RT" and "FAV" are removed.

Handling of Emojis and Smileys: Emojis and smileys are removed from the tweets.

Removal of Punctuation: Punctuation marks are removed.

Removal of Special Characters: Special characters are also removed.

Removal of Numbers: Numeric characters are removed from the tweets.

So, in summary, the preprocessing step involves cleaning and modifying the original tweets by performing the actions mentioned above to prepare the text data for further analysis and summarization.

Implementation:

Data Collection: Gather a dataset of online comments or messages, some of which contain abusive or harmful content, and some that are non-abusive (normal).

Text Preprocessing: Preprocess the text data by cleaning it, removing special characters, tokenizing, and converting to lowercase.

```
import nltk

from nltk. tokenize import word_tokenize

nltk. download('punkt')

def preprocess_text(text):

    text = text.lower()

    tokens = word_tokenize(text)

    tokens = [token for token in tokens if token. isalpha()]

    return ' ‘. join(tokens)
```

Creating Contextualized Representations: Use a pre-trained contextualized embedding model like BERT to convert the pre-processed text into contextualized representations.

```
from transformers import BertTokenizer, Bert Model

import torch

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

```
model = BertModel.from_pretrained('bert-base-uncased')

text = "This comment contains abusive language."

# Tokenize and encode the text

input_ids = tokenizer.encode(text, add_special_tokens=True)

input_ids = torch.tensor(input_ids).unsqueeze(0)  # Add batch dimension

# Obtain the contextualized representation

with torch.no_grad ():

    outputs = model(input_ids)

contextualized_representation = outputs[0]
```

Building a Classifier: Train a machine learning classifier, such as a logistic regression or a neural network, using the contextualized representations as features.

```
from sklearn. model_selection import train_test_split

from sklearn. linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(contextualized_representations, labels, test_size=0.2,
random_state=42)

# Train a logistic regression classifier

classifier = LogisticRegression()

classifier.fit(X_train, y_train)

# Predict on the test set

y_pred = classifier.predict(X_test)

# Evaluate the classifier

accuracy = accuracy_score(y_test, y_pred)
```

Model Evaluation: Evaluate the classifier's performance using appropriate metrics, such as accuracy, precision, recall, and F1-score.

Deployment: Once model performs well, you can deploy it to detect online abuse in real-time by providing it with the contextualized representation of user-generated content.

2.2Class-dependent Lower Dimensionality Statistical Embedding (LDSE) representation:

used for capturing stylistic features in user tweets and estimating the probability of term occurrences in two classes: hate and non-hate spreaders. Process to break data in classes

Preprocessing: Before applying the Char-LDSE representation, the user tweets are preprocessed. Preprocessing typically involves tasks such as text cleaning, lowercasing, removing punctuation, and possibly stemming or lemmatization to prepare the text data for further analysis.

Character Ngram Matrix: Once the preprocessing is complete, a character ngram matrix is created from the text data. A character ngram is a contiguous sequence of characters of a specified length (e.g., 2-grams or 3-grams). This matrix likely represents the frequency or presence of these character ngrams in each tweet.

TFIDF Weighting: TFIDF (Term Frequency-Inverse Document Frequency) weighting is applied to the character ngram matrix. TFIDF is a statistical measure used to evaluate the importance of a term within a collection of documents. It assigns higher weights to character ngrams that are rare in the entire corpus but frequent in a specific document (tweet), indicating their significance.

LDSE (Lower Dimensionality Statistical Embedding): LDSE is calculated using the TFIDF matrix. LDSE, in this context, likely refers to a technique that reduces the dimensionality of the TFIDF-weighted character ngram matrix while retaining relevant stylistic features and information about term occurrences.

Weighted Probability of Terms per Class: After obtaining the LDSE representation, the next step involves calculating the weighted probability of terms per class. This step likely aims to determine the likelihood of specific character ngrams or terms appearing in tweets from the two classes (hate and non-hate spreaders). This information can be useful for identifying distinguishing features between the two classes.

Overall, this process appears to be a feature extraction and representation approach designed to capture stylistic and term occurrence differences between user tweets belonging to hate and non-hate spreader categories.

Implementation

```
import numpy as np

from sklearn. feature_extraction.text import TfidfVectorizer

from sklearn. decomposition import PCA

from sklearn.datasets import fetch_20newsgroups

 # Example dataset

# Load a sample dataset (20 newsgroups) for illustration purposes

categories = ['sci.space', 'rec.autos']

data = fetch_20newsgroups(subset='train', categories=categories, remove=('headers', 'footers', 'quotes'))

documents = data.data

labels = data.target

# TF-IDF vectorization

tfidf_vectorizer = TfidfVectorizer()

tfidf_matrix = tfidf_vectorizer.fit_transform(documents)

# Separate data into two classes

class_0_indices = np. where (labels == 0) [0]

class_1_indices = np. where (labels == 1) [0]

class_0_tfidf = tfidf_matrix[class_0_indices]

class_1_tfidf = tfidf_matrix[class_1_indices]

# Apply PCA separately for each class

n_components = 10

# Adjust the number of components as needed

pca_class_0 = PCA(n_components=n_components)

ldse_class_0 = pca_class_0.fit_transform(class_0_tfidf.toarray())

pca_class_1 = PCA(n_components=n_components)
```

```
ldse_class_1 = pca_class_1.fit_transform(class_1_tfidf.toarray())

# Output the LDSE representations

print("LDSE Representation for Class 0:")

print(ldse_class_0)

print("\nLDSE Representation for Class 1:")

print(ldse_class_1)
```

3.Conclusion

"Class-dependent Lower Dimensionality Statistical Embedding (LDSE)" and "Contextualized Representation" are two distinct techniques used in natural language processing (NLP) and text analysis, each serving specific purposes:

3.1Class-dependent Lower Dimensionality Statistical Embedding (LDSE): LDSE is primarily used for dimensionality reduction in text data with a focus on class-specific distinctions. It aims to reduce the number of features while preserving relevant information for different classes or categories in a classification problem. This can improve the efficiency and effectiveness of text classification or clustering tasks.

Operation: LDSE methods, such as PCA or t-SNE, transform high-dimensional text representations (e.g., TF-IDF, word embeddings) into lower-dimensional spaces. The dimensionality reduction is performed separately for different classes, allowing for class-dependent distinctions in the reduced representations.

3.2 Contextualized Representation:

Contextualized representations are designed to capture the meaning and context of words or tokens within a text, considering the entire document. They are especially useful for tasks that require understanding nuances, word sense disambiguation, and context-aware text analysis.

Operation: Contextualized models, like BERT and GPT-3, use deep neural networks, typically based on transformer architectures, to generate embeddings for words or tokens in a sentence. These embeddings consider the entire context of the text, resulting in representations that vary depending on a word's position in the sentence.

In practice, the choice between these techniques depends on the specific NLP task and goals:

Use LDSE when you want to reduce the dimensionality of text data to improve efficiency and interpretability in classification or clustering tasks. It's particularly valuable when you have distinct classes or categories to differentiate.

Use Contextualized Representations when you need to understand the nuanced meaning and context of words or tokens within a text. Contextualized embeddings are beneficial for tasks such as sentiment analysis, question answering, machine translation, and more, where understanding context is crucial.

Ultimately, the choice between LDSE and contextualized representations depends on the requirements of your specific NLP application and the nature of the text data you are working with.

4.References

F. Rangel, G. L. D. L. P. Sarracén, B. Chulvi, E. Fersini, P. Rosso, Profiling Hate Speech Spreaders on Twitter Task at PAN 2021

J. Risch, R. Krestel, Toxic comment detection in online discussions

J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation

T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space

P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information

J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding:

Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners

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