Hate Speech in Social Media

Hate and Offensive Language Identification (Subtask : H2)

Group: 25

PROBLEM STATEMENT

Social media has seen immense growth over the past decades and, with it, a subsequent increase in hate speech. Hate speech includes offensive language that can be derogatory and insulting and is usually directed to hurt someone's sentiment. Social media platforms need to identify hate speech without imposing any rigid censorship. NLP and AI are being used to tackle this problem and automate flagging hate speech.

DATASET DESCRIPTION

- Our project uses the HASOC Subtask A dataset and aims to identify hate speech in the dataset as accurately as possible.
- The tweets in the dataset are a combination of both hindi and english words, written in Devanagri and English respectively.
- There were emojis, URL's, usernames and hashtags in the tweets.
- The tweets had offensive words in both the languages.
- The tweets are classified in two classes, HOF for hate and offensive tweets, NOT for not offensive or hate tweets.

#Chinesevirus To hell with this Chinese Virus
The situation in India is incredibly heartbreaking. Their COVID # s continue to climb. #IndiaCovidCrisis https://t.co/EDIIh79RcD
@Ravish_sir1 @Jalebi_Bai_ तुम कमीने जैसे लोग योगी के नाम लेने लायक नहीं हो
This didn't age well #ResignModi #ModiAbandonedIndia #ModiKaVaccineJumla #ModiMadeCrisis https://t.co/ws4QcYguYe
गांधीगिरी सिंड्रोम #BengalBurning https://t.co/DZv5Ysr01N

RELATED WORKS

HateCheckHin

A lot of research has been done in the field of hate speech detection using AI and NLP. The paper evaluates existing models for Hindi hate speech detection and identifies the possible weaknesses in the model. It uses mBERT as a base model fine-tuned on two datasets and uses Hindi as the base language. HateCheckHIn uses 28 functionalities and 6 different multilingual settings to identify any flaws in existing models. The paper concludes that the current models work poorly for multilingual test cases.

Hindi-English Hate Speech Detection: Author Profiling, Debiasing, and Practical Perspectives

The paper aims to model and analyze low-resource code-switched Hinglish hate speech on social media and text for graph-based author profiling based on linguistic homophily. It uses two datasets for testing and incorporates homophily with Node2Vec, and observes a significant increase in performance for the two datasets in comparison to existing models with bais elimination.

04

METHODOLOGY

Pre-processing

1. The tweets with a few words in Hindi or entirely in Hindi were translated to English using the python module googletrans.
2. Then we used tweet-preprocessor to deal with URLs, Usernames, Mentions, Reserved Words, Emojis etc.
3. Using NLTK Library, the remaining preprocessing was done. In this step, we removed digits, lower-cased the entire text, removed the punctuations and even lemmatized the tokens.

05

METHODOLOGY

Non-Contextual Embeddings

1. Vectorizer -Tf_IDF and Classifier: Random forest classifier
2. Vectorizer -Tf_IDF and Classifier: Adaboost algorithm
3. Vectorizer -Tf_IDF and Classifier: SVM

Contextual Embeddings

1. Vectorizer -all-mpnet-base-v2 and Classifier: SVM
2. Vectorizer -distil-roberta and Classifier: SVM
3. Vectorizer -BERT and Classifier: GBDT

05

METHODOLOGY

Hyper-parameters

While using pre-trained model 'distilroberta-base', we used [do_lower_case=True ,max_seq_length=512] in the embedding layer. In the dense layer we used, [out_features=512, activation_function=nn.GELU()].

While using MLP, hyperparameters used were:[hidden_layers=(512,128,16), activation='tanh', optimizer='adam', learning_rate='adaptive', learning_init=0.001, alpha=0.0001, max_iter=1000].

EXPERIMENTAL RESULTS

Table 1: Macro F1 score comparisons

Vectorizer	Classifier	F1-macro
Tf_IDF	Random forest	0.48
Tf_IDF	ADA	0.32
Tf_IDF	SVM	0.50
all-mpnet-base-v2	SVM	0.76
distil-roberta	SVM	0.74
BERT	GBDT	0.72
all-mpnet-base-v2	**SVM**	**0.77**
all-mpnet-base-v2	MLP	0.70
xlm-roberta-base	SVM	0.66

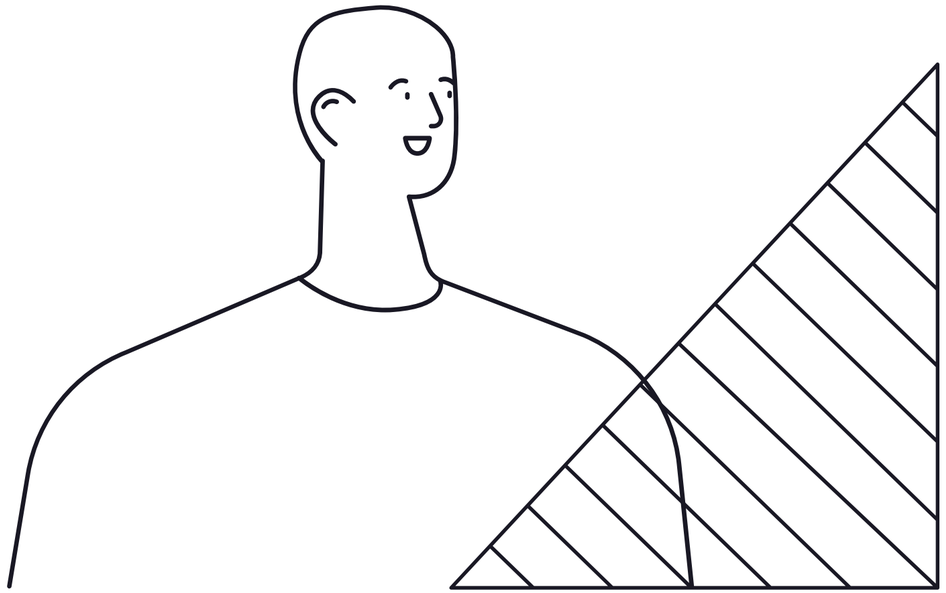
07

Non-contextual Embeddings

1. Vectorizer -Tf_IDF and Classifier: Random forest classifier
Tf_IDf vectorizer gives non-contextual sentence embedding. This causes the inefficient mapping of vectors with the labels {"HOF", "NOT"}.

2. Vectorizer -Tf_IDF and Classifier:SVM(Support Vector Machine)
Using SVC classifier gives the best output from all the different classifiers used throughout the project. Therefore, the output F1-score increased from the previous one.

EXPERIMENTAL RESULTS



Contextual Embeddings

Vectorizer -all-mpnet-base-v2 and Classifier: SVM(Support Vector Machine)

We can see that this pair is written twice in the table with two different sets of accuracies. The original accuracy took a jump from earlier high 50(from non-contextual) to 76. The reason being that "all-mpnet-base-v2" is a contextual sentence transformer and hence gives a better relation between the tags and the vectors. Further, this score was improved when the data was pre-processed in a better way. Earlier we had removed all the hashtags, which took away a lot of semantics, but now along with some other modifications, we kept the hashtags, and hence obtained a score of 77.

Vectorizer -distil-roberta and Classifier: SVM(Support Vector Machine)

Using distil-roberta still was better than any non-contexual embedding, however it didn't perform as well as all-mpnet-base-v2. This was expected behaviour as described in the sentence-transformers documentation.

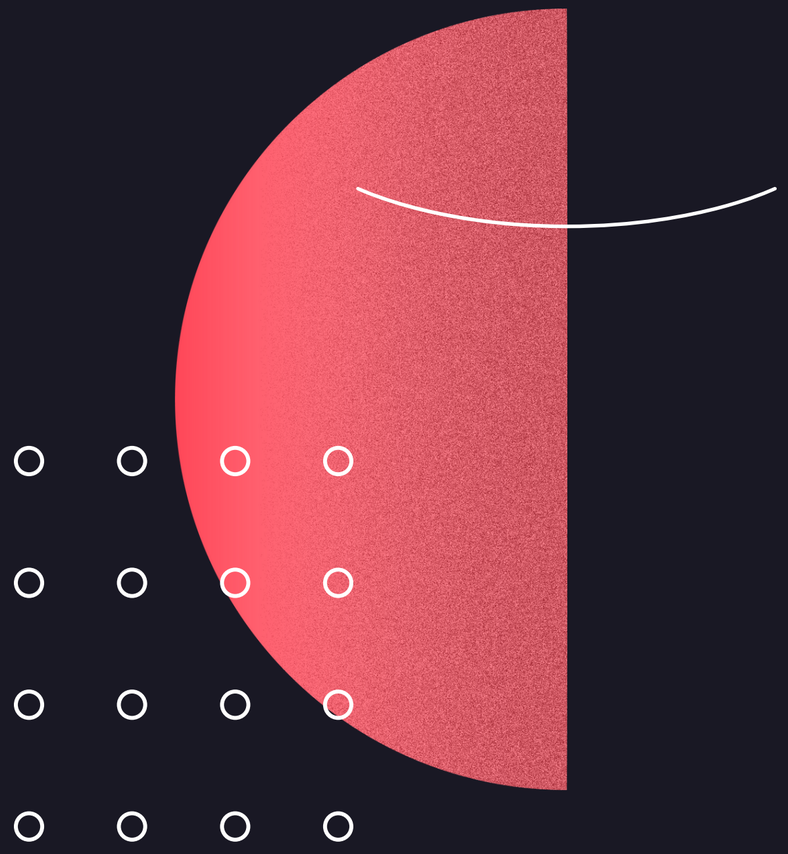
Vectorizer xlm-roberta-base and Classifier: SVM(Support Vector Machine)

Contextual embeddings without pre-translation

Here we obtained a score of 66. Instead of translation from Hindi to English using google translate, we passed the original data into the Cross-lingual language model. This however performed poorly as compared to previous models. Probably indicating the google-translate's machine translation is better than this direct cross-lingual classification.

ANALYSIS AND FINAL COMMENTS

From the different models and techniques we have used so far, we observed that pre-processing the data by (1)Translating Hindi tweets to English using Google translate, (2)Adding the keywords after segmenting¨" hashtags, (3)removing stop words, URLs, emo-jis, and tags produced better features. Further-more, using contextual sentence transformer, specifically, "all-mpnet-base-v2," gave the best feature vectors. From a range of ML and DL classifiers used, SVC gave the best accuracies and F1-score.



CONTRIBUTIONS

Harjeet(2020561) and Charvi(2020045)

Pre-processing and making report

Aaditya(2020552) and Chetan(2020046)

Resource reading and running different models