Homophobia/Transphobia Detection in social media comments LT-EDI@EACL 2024

State of the art

1. Introduction

Online users are negatively impacted by the growing amount of offensive information on social media platforms. Homophobia and transphobia are terms used to describe feelings of fear, hatred, tension, or reservations towards lesbian, gay, transgender, or bisexual people.

In recent years, there has been an increase in the concern over homophobic and transphobic speech, which is a form of unpleasant language that can be best described as hate speech aimed at LGBT+ individuals. Online homophobia and transphobia are serious social issues that have the potential to negatively impact online spaces and make LGBT+ individuals feel unwelcome while also working to undermine inclusion, diversity, and equality.

The increasing amount of online content has made it a serious issue for online communities. Over the past ten years, social media sites like Facebook, YouTube, and Twitter have become hubs for the global problem known as online abusive language. LGBT+ vulnerable individuals find it considerably more upsetting nowadays.

2. Models and Techniques

State-of-the-art models and techniques for detecting homophobia and transphobia in social media comments leverage a combination of traditional machine learning and advanced NLP approaches. Notable approaches include:

- BERT-based Models:
    - Bidirectional Encoder Representations from Transformers (BERT) and other transformer-based models have demonstrated success in capturing contextual information.
    - Fine-tuning pre-trained models, such as BERT, for specific hate speech detection tasks shows promise as a base model.
- Feature Extraction Techniques:
    - TF-IDF, count vectorizer, fasText for training multi-class homophobia/transphobia classifiers.

    - Both machine learning (SVM, Naive Bayes, Logistic Regression, Random Forest, CNN, LSTM, KNN, Decision trees) and deep learning (BiLSTM and mBERT) approaches are explored.
- Multilingual Embeddings:
    - Models like mBERT and XLM-R, designed for handling various languages, offer potential for cross-lingual transfer tasks.

Different deep learning-based models are implemented to detect homophobia and transphobia in comments, therefore there are multiple directions which we may explore before deciding on a definite workflow - either focus on one model and aim to make improvements on it → a BERT-based transformer model, for example, or build multiple models and experiment with multiple approaches, compare them and analyze which one performs best.

3. Proposed Approach

- A potential approach involves the implementation and evaluation of various models:
- Implementation of Deep Learning Models:
    - Utilize models (SVM, Naive Bayes, Logistic Regression, Random Forest, CNN, LSTM, KNN, Decision trees) with GloVe embeddings for data in all languages.
- Implementation of Transformer-based Models:
    - Explore transformer models (mBERT, IndicBERT, RoBERTa, HateBERT, RoBERT-base) for data in all languages.
- Performance Evaluation:
    - Evaluate the implemented models on the provided dataset using standard metrics.
    - Pretrain and fine-tune models in English and test on various languages (Hindi, Tamil, Telugu, Kannada, Gujarathi, Malayalam, Marathi).

4. Alternative Approach

Consider analyzing multiple papers on the subject, focusing on performance and techniques. Implement a chosen approach, fine-tune hyperparameters, and aim for good performance in languages not studied extensively. (We could not really find previous work on detecting homophobia and transphobia on social media comments in Malayalam language, for example).

5. Related Work

Explore relevant literature, including:

- https://aclanthology.org/2022.ltedi-1.55/: Multilingual tests, closely related to the task.
- https://arxiv.org/ftp/arxiv/papers/2304/2304.01241.pdf: Deep learning experiments, potentially suitable for the multilingual task.
- https://aclanthology.org/2022.ltedi-1.19.pdf: RoBERTa approach, requiring analysis for multilingual applicability.
- https://link.springer.com/article/10.1007/s41060-023-00400-0: Additional insights into related work.

6. Additional Tasks

Other tasks that we might need to perform:

- preprocess the comments for special characters, stopwords, emojis, and punctuation removal (possibly using NLTK library)
- extract the features, tokenize all the sentences and map the tokens to their word IDs.
- for every sentence in the dataset: tokenize the sentences, add a token to the start, append a separator token to the end, map the token to their IDs, pad or truncate the sentenced to a predefined max length etc.
- After obtaining the embeddings: feed these embeddings for pretraining the (XLM?) *BERT-based model*(something BERT-based for sure) - we need to choose something that aims at cross-lingual transfer tasks for pre-trained multilingual language models. The model needs to perform well on low resource languages like Malayalam (without sacrificing per-language performance!).
- Maybe we can compare the results to strong monolingual models (GLUE - an example) in order to assess whether our results are relevant.

7. Conclusion

This state-of-the-art document outlines a comprehensive plan for detecting homophobia and transphobia in social media comments using a combination of traditional machine learning and advanced NLP techniques. The proposed models and approaches aim to address the multilingual nature of the task, contributing to a more inclusive and effective detection system.

About our solutions

The tasks of this competition - basically doing a model separately for each language (English, Spanish, Tamil, Gujarathi, Kannada, Malayalam, Telugu, Hindi, Tulu, Marathi) for classifying youtube comments in that specific language into either Homophobic, Transphobic or Non anti LGBT+ content. - the metric used for testing - **macro F1**.

Idea: instead of making 9 - 10 separate models for each language, make only one model that can make predictions given any of those 10 languages as test input. For this, we needed one single training dataset, not 10 separate ones. But - the main issue - imbalance!
The trick that most probably improved a lot the performance - as most datasets were heavily imbalanced - augmentation - back translation, sometimes combined with oversampling.

What I've done:

1. Data cleaning - re-naming and cleaning all datasets. The same naming conventions applied to the three classes in all files. The same column names, etc.
2. Data preparation - prepared the datasets for further tasks
3. Exploratory data analysis - this basically involved analyzing the degree of imbalance between classes.
4. Preparing the test sets - the same as train sets…
5. Augmentation techniques - the most important step. I made 3 datasets, each one of them resulting from a different measure of augmentation/ a different way of applying the techniques. Basically what I did and improved a lot the results - back translation - translate the comments from underrepresented categories (Homophobia and Transphobia) into another language, then translate the resulted text back to the original language - this way, we introduce more samples for underrepresented categories, and getting a bit of variety in text formulation at the same time. After one round of back translation, I performed back translation again, on the doubled text, depending on the imbalance level of a certain dataset. I used the translate_v2 API from Google Cloud - this supports translation for many languages, including the ones from our task.
6. Oversampling - After back translation, I performed oversampling: either at language level, or at general level. At language level: **Dataset 1**: doing back translate to a certain degree, then, doing oversampling on underrepresented categories until we get

100% balance between these 3 classes - same number of samples - but only at language level (we have for example 3000 samples with language A with 1000 samples of each category, then we have 6000 samples of language B with 2000 samples of each language and so on. And, we concatenate all languages, and then shuffle the rows, and we get the dataset. **Dataset 2**: doing only back translation - back translate as many samples as necessary in order to get the relative balance explained at Dataset 1 (instead of oversampling what remains, do only back translation until we get balance between classes at language level).

Dataset 3: Do more back translation: back translate as many samples as necessary in order to get the relative balance explained at Dataset 1 (instead of oversampling what remains, do only back translation until we get balance between classes at language level). Afterwards, obtain a general balance - do (uniform) oversampling on the languages which are underrepresented and obtain the same number of samples for each of those 9 languages - do this in an uniform way so that the balance between the categories remains. Managed to compose 3 separate training datasets as described above - analyzed the performance on each one of them.

Used huggingface framework - pretrained model -

`distilbert-base-multilingual-cased`

Trained for 15 epochs. - chose: the final model + one checkpoint -> Total: 6 models.

Models: Model 1 - a bit of back translation, then oversampling - balance only at language level (76k rows). After 15 epochs.

Model 1 - c: checkpoint for model 1 - epoch 5.

Model 2: full back translation, without any oversampling afterwards so balance only at language level (76k rows). After 15 epochs.

Model 2 - c: checkpoint for model 2 - epoch 8.

Model 3: full back translation, then oversampling at general level. 120k+ rows. After 10 epochs.

Model 3 - c: checkpoint for model 3 - epoch 6.

The table below summarizes the macro F1 score on all test sets, the last column being the score from the 1st place in the competition:

Results:

English - 1st place

Malayalam - 3rd place

Tamil - 3rd place

Marathi - 2nd place

Spanish - 2nd place

Hindi - close to 7th place

Telugu - 1st place

Kannada - 6st place

Gujarathi - 2nd place

Tulu - (not trained on Tulu dataset) - 3rd place

Lg/Mdl	1	1-c	2	2-c	3	3-c	1stPlace
English	**0.611**	0.515	0.571	0.570	0.601	0.589	0.496
Malayalam	0.885	0.851	0.868	**0.895**	0.851	0.810	0.942
Tamil	0.836	0.826	0.829	**0.844**	0.808	0.808	0.880
Marathi	0.5680	0.544	0.561	**0.5684**	0.560	0.553	0.626
Spanish	**0.570**	0.558	0.549	0.546	0.543	0.554	0.582
Hindi	0.320	0.320	**0.324**	0.323	0.318	0.319	0.458
Telugu	**0.972**	0.964	0.966	0.962	0.969	0.966	0.971
Kannada	0.931	0.931	**0.934**	0.933	0.933	0.928	0.948
Gujarathi	0.962	0.957	**0.965**	0.960	0.953	0.961	0.968
Tulu	**0.464**	0.337	0.350	0.331	0.318	0.338	0.707

English			
Team name	**Run**	**M_F1-score**	**Rank**
dkit	Run1	0.496	1
MUCS	Run2	0.493	2
KEC_AIDS	-	0.466	3
CUTN_CS_HOMO	BERT	0.457	4
SCaLAR	Run3	0.438	5
MEnTr	-	0.407	6
Hypnotize	-	0.384	7
KEC_AI_NLP	Run1	0.369	8
quartet	-	0.347	9
cantnlp	Run1	0.323	10
MasonTigers	-	0.323	10

Malayalam			
Team name	**Run**	**M_F1-score**	**Rank**
CUTN_CS_HOMO	MuRIL	0.942	1
Hypnotize	-	0.909	2
bytellm	-	0.891	3
KEC_AI_NLP	Run1	0.883	4
quartet	-	0.877	5
MUCS	Run3	0.870	6
cantnlp	Run2	0.775	7
MEnTr	-	0.744	8
MasonTigers	-	0.505	9

Tamil			
Team name	**Run**	**M_F1-score**	**Rank**
Hypnotize	-	0.880	1
MUCS	Run3	0.860	2
bytellm	-	0.801	3
MEnTr	-	0.746	4
MasonTigers	-	0.512	5
quartet	-	0.483	6
KEC_AI_NLP	Run1	0.315	7

Marathi			
Team name	**Run**	**M_F1-score**	**Rank**
Hypnotize	-	0.626	1
MUCS	Run2	0.537	2
MEnTr	-	0.488	3
MasonTigers	-	0.438	4
cantnlp	Run1	0.433	5
quartet	-	0.391	6

Spanish			
Team name	**Run**	**M_F1-score**	**Rank**
MEnTr	-	0.582	1
MUCS	Run3	0.532	2
MasonTigers	-	0.499	3
KEC_AI_NLP	Run1	0.369	4

Hindi

Team name	Run	M_F1-score	Rank
MUCS	Run2	0.458	1
SCaLAR	Run1	0.410	2
Hypnotize	-	0.403	3
cantnlp	Run1	0.326	4
quartet	-	0.326	4
MasonTigers	-	0.326	4
MEnTr	-	0.325	5

Telugu

Team name	Run	M_F1-score	Rank
Hypnotize	-	0.971	1
MasonTigers	-	0.971	1
MEnTr	-	0.960	2
byteLLM	-	0.959	3
MUCS	Run1	0.958	4
SCaLAR	Run1	0.911	5
quartet	-	0.891	6
KEC_AI_NLP	Run1	0.369	7

Kannada			
Team name	**Run**	**M_F1-score**	**Rank**
MUCS	Run2	0.948	1
Hypnotize	-	0.946	2
MasonTigers	-	0.945	3
cantnlp	Run1	0.943	4
MEnTr	-	0.935	5
bytellm	-	0.922	6
SCaLAR	Run1	0.903	7
quartet	-	0.887	8

Gujarathi			
Team name	**Run**	**M_F1-score**	**Rank**
Hypnotize	-	0.968	1
cantnlp	Run1	0.962	2
MEnTr	-	0.960	3
MUCS	Run2	0.958	4
MasonTigers	-	0.935	5
quartet	-	0.893	6

Tulu			
Team name	**Run**	**M_F1-score**	**Rank**
MEnTr	Run1	0.707	1
MUCS	Run2	0.620	2
MasonTigers	Run1	0.452	3
cantnlp	Run1	0.452	3

We’ve also experimented with a simpler model to put in perspective the difference. We’ve used Multinomial Naive Bayes with a Bag of Words representation of the text.
The results were the following:

- When testing on a dataset containing all languages - F1 score of 0.76

When testing over individual languages:

English - F1 score 0.45 - 5th place

Malayalam - F1 score 0.55- 9th place

Tamil - F1 score 0.57 - 5th place

Marathi - F1 score 0.40 - 6th place

Spanish - F1 score 0.535 - 2nd place

Hindi - F1 score 0.30 - close to 7th place

Telugu - F1 score 0.81 - 7th place with a big lead

Kannada - F1 score 0.86 - close to 8th place

Gujarathi - F1 score 0.86 - close to 6th place

Tulu - F1 score 0.31 - pretty big difference between this and the last place