Proceedings of the OSACT 2022 Workshop @LREC2022, pages 214–217
Marseille, 20 June 2022

aiXplain at Arabic Hate Speech 2022: An Ensemble Based Approach to Detecting Offensive Tweets

Salaheddin Alzu'bi, Thiago Castro Ferreira , Lucas Pavanelli, Mohamed Al-Badrashiny
aiXplain Inc.
Los Gatos, CA
(salah.alzubi, thiago, lucas.pavanelli, mohamed)@aixplain.com

Abstract

Abusive speech on online platforms has a detrimental effect on users' mental health. This warrants the need for innovative solutions that automatically moderate content, especially on online platforms such as Twitter where a user's anonymity is loosely controlled. This paper outlines aiXplain Inc.'s ensemble based approach to detecting offensive speech in the Arabic language based on OSACT5's shared **sub-task A**. Additionally, this paper highlights multiple challenges that may hinder progress on detecting abusive speech and provides potential avenues and techniques that may lead to significant progress.

Keywords: Offensive Language Detection, Arabic Shared Task, Abusive Language Detection

1. Introduction

The presence of abusive speech in online communities poses a detrimental effect to many users' mental health. This is particularly apparent in interactive platforms such as Twitter, where a user's anonymity is loosely moderated. This phenomenon warrants the need for models that automatically detect and handle abusive language; while solutions to this problem in the English language have been proposed, there is room for improvement for the Arabic language.

This paper presents aiXplain Inc.'s ensemble based approach to detecting offensive speech in Arabic for the OSACT5 shared sub-task A (Hassan et al., 2020). In particular, this paper outlines the individual approaches that were used in tackling this task and offers a final system architecture based on a combination of these models. Additionally, this paper offers insights into the challenging aspects of classifying offensive speech in Arabic and potential future directions that may bear fruit in tackling this problem.

2. Data & Task Description

In this section, we provide an overview of the data and a description of the sub-tasks available.

2.1. Data

The dataset provided for OSCAT5 Arabic hate speech task consists of 12698 annotated tweets that serve as the largest annotated database for the Arabic community. Each tweet is independently judged by 3 annotators that assign the following labels to them: offensive/not-offensive; hate-speech/not-hate-speech; type of hate-speech: race, religion, ideology, disability, social class, and gender; additionally, each tweet is labeled as vulgar/not-vulgar and violent/not-violent. The dataset is split into a training set, development set and testing set with the following distribution:

Train	Development	Test
8887	1270	2541

Table 1: OSACT5 Arabic Hate Speech Data Distribution

It is worth emphasizing that the dataset is heavily imbalanced for both available sub-tasks, with the second & third sub-tasks (hate speech classification) being more severe than the first.

Subtask	Label 0 (%)	Label 1 (%)
A	65	35
B	90	10

Table 2: Subtask Label Distribution

2.2. Shared Sub-tasks

This section provides an overview of the three different sub-tasks that are available. We would like to emphasize the fact that our submission involves participation in **Subtask A only**.

2.2.1. Subtask A: Offensive Speech Detection

The first subtask involves detecting whether a tweet contains offensive speech or not. Offensive speech is defined as any kind of implicit or explicit insults or attacks against individuals or groups.

2.2.2. Subtask B: Hate Speech Detection

The second sub-task involves detecting hate speech. Hate speech is defined as any kind of implicit or explicit speech that targets an individual's, or groups', race, religion, ideology, disability, social class, or gender.

2.2.3. Subtask C: Hate Speech Detection

The third sub-task is the same as the second sub-task, however, the problem is posed as a multi-class classification problem. The primary aim of this sub-task is to perform fine-grained classification of tweets that contain hate-speech into one of the six classes mentioned in the section above.

3. Approach

This section includes the primary approach we used to achieve the best results. Our approach consists of three main steps: Augmentation, Pre-processing and passing the data through an ensemble[1].

3.1. Preprocessing

This section discusses the pre-processing techniques that we apply on tweets. Our pre-processing technique involves dealing with text and emojis separately.

3.1.1. Textual Pre-processing

For the textual part of the tweet, we apply the following transformations sequentially on each tweet:

1. Remove URLs and mentions
2. Remove diacritics and tatweel
3. Remove punctuation

This simple approach ensures that the data remains faithful to the original distribution while removing noisy signals in the tweet.

3.1.2. Emoji Pre-processing

(Mubarak et al., 2022) show that emoji's provide invaluable information to detect a tweet's sentiment. Since most models are not directly trained to handle tweets, we translate emojis in a tweet to Arabic using (Junczys-Dowmunt et al., 2018) English to Arabic model.For some emojis, we infer their intended meaning (from how they are usually used in the region's context) and provide their translation using our expertise in the Arabic language and the colloquial dialect used. Additionally, we extract the relevant emojis from each tweet and use a classifier to predict their sentiment individually. These predictions are used as additional high-level features to other classifiers.

3.2. Data Augmentation

Since deep-learning models such as BERT (Devlin et al., 2018) are data hungry, a large amount of data from each class is required for any significant learning to occur. This section outlines the two methods that were used to augment the offensive dataset.

3.2.1. Semi-Supervised Learning

We use recent developments in semi-supervised learning to augment the classes with less support (in this case: offensive class). We fine-tune a pre-trained AraBERTv0.2-Twitter-base (Antoun et al., 2020) model on detecting offensive speech; this model is then used to classify a large set of tweets that have been scraped from external sources. We select the tweets that the model predicts as offensive with high confidence.

[1]https://github.com/aixplain/arabic-hate-speech

3.2.2. Contextual Augmentation based on Semantic Similarity

We use the Python package NLPAug (Ma, 2019) to augment the tweets. This technique consists of feeding surrounding words to AraBERTv0.2-Twitter-base (Antoun et al., 2020) to find out a semantically similar word that is suitable for augmentation. We run this augmentation for every offensive example, randomly substituting 30% of the words with the similar one and, thus, generating a new example. We apply this technique till we double the number of examples in the offensive class making the class distribution balanced. Based on our empirical observations, we have found that changing around 2-3 words in a tweet preserves the overall meaning whilst adding capturing a broader spectrum of the negative sentiment.

3.3. System Overview

This section provides a detailed overview of the approach that we use to make the final predictions on tweets. Our system's architecture involves feeding the predictions of an ensemble of classifiers combined with relative high-level features to a final meta-learner yielding a binary label of "OFF" to represent offensive or "NOT_OFF" to represent unoffensive speech.
Each of the classifiers in the ensemble consist of a final linear layer the following pre-trained model as a backbone:

- AraBERTv0.2-Twitter-large (Antoun et al., 2020)
- Mazajak 250M CBOW pre-trained embeddings (Abu Farha and Magdy, 2019)
- Character level N-gram + word level N-gram TF-IDF embeddings (Takase et al., 2019)
- MUSE (Conneau et al., 2017)

Additionally, we use our Emoji model to extract additional high level features from each tweet.
The predictions from the aforementioned models are then concatenated into a final vector which is fed into either a final XGBoost model or a linear layer to produce the final binary prediction.

3.4. Models

This section contains an overview of the models that we use as part of our ensemble.

3.4.1. AraBERTv0.2-Twitter-large

AraBERTv0.2-Twitter (Antoun et al., 2020) is a pre-trained transformer model that is based on Google's BERT (Devlin et al., 2018) model. Similar to BERT the model is pre-trained on an MLM task using a collection of 60M arabic tweets. This particular model contains emojis as part of its vocabulary making it suitable for this task. We use HuggingFace's API to fine-tune our model using Adam (Kingma and Ba, 2014) optimizer and a learning rate of 1e-5.

3.4.2. Mazajak Pre-trained Embeddings

Mazajak embeddings (Abu Farha and Magdy, 2019) are the largest available embeddings for the arabic langauge trained on 250M tweets. Mazajak embeddings were created by training a Continuous Bag-of-Words (CBOW) model to yield a 300-dimensional contextual vector for each word in the corpus. In our model, we use the mazajak embeddings for each word and apply average pooling to produce a final 300-dimensional vector that is fed into a linear layer to give a prediction.

3.4.3. Character + Word level N-gram TF-IDF Embeddings

In this model, we combine the tri-gram character level tf-idf embeddings of tweets with the bi-gram word level tf-idf embeddigs. We believe that lexical level features such as character level and word level embeddings add an extra dimension to the learning of our ensemble classifier.

3.4.4. MUSE

This model utilizes the word embeddings provided by FAIR's Multilingual Universal Sentence Encoder (Conneau et al., 2017). Given a tweet, we feed the pre-processed data to MUSE to generate a 512 word embedding vector; this vector is then fed into a logistic regression layer to provide a final prediction.

3.4.5. Emoji Score

Emoji score is the model we use to extract additional high-level features from the emoji's present in a tweet. In particular, we assign a score to each emoji based on how many times it is used in offensive tweets and how many times it is present in non-offensive ones. For each tweet, we aggregate the emoji-score of each emoji into a final score representing the emoji-score.

An additional approach we experimented with involved a bag-of-words model that calculated the offensiveness of a tweet based on the emoji scores in a tweet.

4. Model Evaluation

This section presents and discusses the performance of our models on the development set. We also present the final evaluation scores on the test set provided.

4.1. Dev Set Results

The development set is used as a benchmark for our model's generalization performance on unseen data; the table below shows the performance of each individual model and different combinations of the models on the most common metrics used to evaluate binary classification.

Model	P	R	Macro F1
Char-tfidf	0.79	0.72	0.74
Word-tfidf	0.75	0.65	0.66
Emoji	0.68	0.56	0.54
Muse	0.73	0.69	0.7
Emoji-Score	**0.68**	**0.55**	**0.51**
AraBERT	**0.84**	**0.83**	**0.84**
Mazajak	0.73	0.64	0.64
Char+word+MUSE	0.79	0.74	0.75
Char+word+MUSE +Emoji	**0.79**	**0.76**	**0.77**
Ensemble of Boldface Models	**0.86**	**0.85**	**0.85**

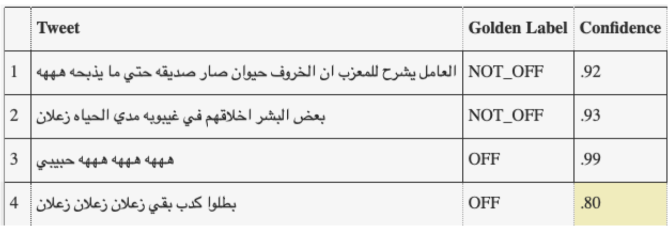
Table 3: Evaluation of different models on the Precision (P), Recall (R) and Macro F1-Score binary classification performance metrics

4.1.1. Results Discussion

The results above show that AraBERT outperforms all the other models by a large margin. We believe that this is due to the fact that AraBERT was specifically trained on a large number of tweets that captured the datas underlying distribution. A surprising result is the poor performance of the Mazajak pre-trained embeddings as they are also trained on 250M tweets, yet the Mazajak embeddings did not seem to represent the tweets in a successful manner. We believe that this may be due to a couple of reasons: (1) the underlying distribution of the data (in this case, the different dialects) is different from the distribution that the Mazajak model was trained on. (2) The pre-processing steps that the authors used are different from the steps that we used leading to a large discrepancy in the performance. Other models such as TF-IDF and MUSE show promise but are not up to par with the best result of AraBERT. Under the aforementioned, the final result which involves ensembling multiple models and high-level features only pushes the final result by .02 f1 points; this indicates that apart from AraBERT, the other feature extraction methods are either insignificant or provide weak signals to the final prediction.

4.1.2. Error Analysis

This section highlights some of the examples that our best model gets wrong and provides some insight as to why the model may be behaving the way it is.



	Tweet	Golden Label	Confidence
1	العامل يشرح للمعزب ان الخروف حيوان صار صديقه حتي ما يذبحه ههههه	NOT_OFF	.92
2	بعض البشر اخلاقهم في غيبوبه مدي الحياه زعلان	NOT_OFF	.93
3	هههه هههه هههه حبيبي	OFF	.99
4	بطلوا كدب بقي زعلان زعلان زعلان	OFF	.80

Figure 1: Examples the best model confidently misclassifies

The above results show that our best model struggles to classify examples with no emojis; this supports the claim that language agnostic indicators such as emojis provide valuable insights to the models' predictions. The model also seems to fail to recognize offensive tweets that when placed in context count as offensive such as example (3).

4.2. Test Set Results

The table below shows our model's final result on the test set. The final models that were submitted involved using different meta-learners as the final classification layer. For the first submission, we use a auto-sklearn to select the best estimators; for the second submission, we use a linear layer as the classification layer.

Model	Acc	P	R	Macro F1
1	**0.864**	**0.852**	**0.847**	**0.849**
2	0.858	0.845	0.84	0.843
Baseline	0.651	0.325	0.5	0.394

Table 4: Accuracy (Acc), Precision (P), Recall (R) and Macro F1-Score of Our Best Models on the Test Set

These results are in line with the results achieved in the development phase. This shows that our best model is able to generalize well to unseen tweets from the same distribution.

5. Challenges & Future Directions

The biggest challenge we faced when attempting to detect offensive tweets was normalizing the dialect of the tweets. Most of the available pre-trained models or preprocessing libraries are trained on MSA or a particular Arabic dialect making a unified approach difficult. This limited our ability to extract relevant features from the tweets in a useful manner; for example, POS tags and NER. This lead us to look for relevant signals in emojis as they are, to a large extent, language agnostic. We believe that exploring emoji's and their relevance to classifying offensive speech in tweets can provide valuable signals to the overall prediction.

6. Bibliographical References

Abu Farha, I. and Magdy, W. (2019). Mazajak: An online Arabic sentiment analyser. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 192–198, Florence, Italy, August. Association for Computational Linguistics.

Antoun, W., Baly, F., and Hajj, H. (2020). Arabert: Transformer-based model for arabic language understanding. In *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, page 9.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2017). Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Hassan, S., Samih, Y., Mubarak, H., Abdelali, A., Rashed, A., and Chowdhury, S. A. (2020). ALT submission for OSACT shared task on offensive language detection. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 61–65, Marseille, France, May. European Language Resource Association.

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Fikri Aji, A., Bogoychev, N., Martins, A. F. T., and Birch, A. (2018). Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia, July. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Ma, E. (2019). Nlp augmentation. https://github.com/makcedward/nlpaug.

Mubarak, H., Hassan, S., and Chowdhury, S. A. (2022). Emojis as anchors to detect arabic offensive language and hate speech.

Takase, S., Suzuki, J., and Nagata, M. (2019). Character n-gram embeddings to improve rnn language models.