Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, pages 71–75
with a Shared Task on Offensive Language Detection.
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

OCAST4 Shared Tasks: Ensembled Stacked Classification for Offensive and Hate Speech in Arabic Tweets

Hafiz Hassaan Saeed[1], **Toon Calders**[2], **Faisal Kamiran**[1]
[1] Information Technology University, Pakistan, {hassaan.saeed,faisal.kamiran}@itu.edu.pk
[2]University of Antwerp, Belgium, toon.calders@uantwerpen.be

Abstract

In this paper, we describe our submission for the OCAST4 2020 shared tasks on offensive language and hate speech detection in the Arabic language. Our solution builds upon combining a number of deep learning models using pre-trained word vectors. To improve the word representation and increase word coverage, we compare a number of existing pre-trained word embeddings and finally concatenate the two empirically best among them. To avoid under- as well as over-fitting, we train each deep model multiple times, and we include the optimization of the decision threshold into the training process. The predictions of the resulting models are then combined into a tuned ensemble by stacking a classifier on top of the predictions by these base models. We name our approach "ESOTP" (Ensembled Stacking classifier over Optimized Thresholded Predictions of multiple deep models). The resulting ESOTP-based system ranked 6th out of 35 on the shared task of Offensive Language detection (sub-task A) and 5th out of 30 on Hate Speech Detection (sub-task B).

Keywords: ESOTP, Arabic Hate Speech, Arabic Offensive Language, Stacked Deep Predictions

1. Introduction

Social media platforms have become a widely-used mode of communication among individuals or groups from diverse backgrounds. With the increasing freedom of expression in user-generated content on these platforms, the menace of offensive and hate speech is also on the rise (Santosh and Aravind, 2019), causing adverse effects on users and society at large (Lee and Kim, 2015). Because of this menace, the identification of offensive or hateful statements towards individuals or groups has become a priority nowadays and many social media companies have already invested millions for building automated systems to detect offensive language and hate speech (Gambäck and Sikdar, 2017).

In this paper, we address the task of offensive language and hate speech detection in the Arabic language by presenting our contributions to two shared tasks (A and B) in OCAST4 2020. The objective in shared subtask A is to identify offensive language whereas the objective in shared subtask B is to identify hate speech in the given tweets. We developed a common methodology for both tasks, and executed the classification pipeline twice, once for each of both subtasks.

As a first attempt, we applied classical text classification techniques including Naïve Bayes, Logistic Regression, Support Vector Machines, and Random Forests based on the traditional encoding of the tweets as TF.IDF vectors. Subsequently, more advanced deep learning techniques using pre-trained word embeddings were applied and compared to the classical techniques. Both approaches were compared empirically, showing superior performance for the deep models.

The superiority of the deep models motivated further exploration in the direction of deep learning. We compared a number of pre-trained word-level embeddings available for Arabic language processing, and in the end, concatenated the two empirically best performing pre-trained word-level embeddings.

Using this combined embedding, several network architectures and ways of pre-processing were tried out, and the resulting models were combined in a tuned ensemble as follows. The different deep networks were trained and optimized several times, saving their predictions for both tasks. Finally, a classifier was stacked on top of these predictions to combine them in one ensemble. The resulting classifier was further fine-tuned, therefore, we name our approach "ESOTP" which stands for *Ensembled Stacking classifier over Optimized Thresholded Predictions of multiple deep models*.

2. Related Work

Hate speech detection has been studied extensively in recent years, especially for highly-resourced languages like English. (Yin et al., 2009) were among the first ones to apply supervised machine learning approaches in hate speech detection. They applied Support Vector Machines to detect harassment in posts from famous social platforms like MySpace. Similarly, (Warner and Hirschberg, 2012) trained a Support Vector Machine classifier on word n-grams and used it to detect hate speech. In recent years, (Waseem and Hovy, 2016) showed that character n-grams are better than word n-grams as predictive features for hate speech detection. Their best performing model was a Gradient Boosted Decision Trees classifier trained on word embeddings learned using LSTMs.

There exists, however, very little literature on the problem of Hate Speech detection in Arabic. Some of the few works are discussed next. (Magdy et al., 2015) collected a large number of Arabic tweets and trained a Support Vector Machine classifier to predict if a user supports or opposes ISIS. (Mubarak et al., 2017) proposed a methodology for the detection of profane tweets by using an automatically created and expanded list of obscene and offensive words. (Haidar et al., 2017) proposed a multilingual system that detects cyberbullying attacks in both English and Arabic texts. They scrapped the data from Facebook and Twitter. The data collected from Facebook was kept for validating the system. Their proposed system was a multilingual cyberbullying detection system and two machine learning models Naive Bayes and Support Vector Machine were used in it. In another related work, (Albadi et al., 2018) prepared the first publicly available Arabic dataset that was especially annotated for religious hate speech detection. They also developed multiple classifiers using lexicon-based, n-gram-based, and deep learning approaches. They found a simple Recurrent Neural Network (RNN) architecture with Gated Recurrent Units (GRU) and pre-trained word embeddings to be the best performing model for the detection of religious hate speech in Arabic. (Mohaouchane et al., 2019) compared multiple deep models including CNN, BLSTM with Attention, BLSTM and Combined CNN-LSTM for detecting offensive language in Arabic. They showed that CNN-LSTM achieved best recall scores whereas CNN achieved highest f1 scores in 5-fold cross validation. Recently, (Chowdhury et al., 2019) proposed ARHNET to detect religious hate speech in Arabic by using word embeddings and social network graphs with deep learning models and improved the classification scores than (Albadi et al., 2018). The overview of OSACT4 Arabic Offensive Language Detection Shared Task is discussed by (Mubarak et al., 2020).

3. Methodology

Starting from pre-processing, we now discuss the overall methodology (classification pipeline) followed for both subtasks in OCAST4 2020.

3.1. Pre-processing

One pre-processing step was already done over the original tweets by the competition's organizers, i.e., mentions of a specific user were replaced with @USER, URLs were replaced with URL, and empty lines with <LF>. We removed all these replaced tokens along with emoticons, emojis, punctuation marks (both Arabic and English), English characters, digits (both Arabic and English) and Arabic diacritics. We then normalized a few Arabic characters like *Hamza*, *Ya*, *Ha*, and *Qaf*, and finally removed a repeating character in the string if it is repeated more than 3 times consecutively. An additional pre-processing step is taken for out-of-word-embeddings-vocabulary (OOWEV) words with the models that use pre-trained word embeddings, which is to split an OOWEV word into 2 tokens (i.e., the first character and the rest of the word) if the first character is *Wa*, *Fa*, or *Sa*. The intuition behind this additional step is that *Wa*, *Fa*, or *Sa* appearing at the beginning of an Arabic word function like a grammatical particle as *Wa* gives added meaning of ("and" or "vow" or "oath"), *Fa* gives added meaning of (result to a previous statement) and *Sa* gives added meaning of (in very near future). This way a few more words are covered from the pre-trained word embeddings.

3.2. Pre-trained Word Vectors

A number of pre-trained word vectors are available for Arabic language processing like FastText (Grave et al., 2018), Word2Vec[1] (Continuous Skip gram trained over Arabic CoNLL17 corpus), AraVec (Soliman et al., 2017), N-Gram and Uni-Gram models, and recent BERT[2] multilingual vectors. We empirically evaluated these available word embeddings based on the given evaluation metric and concatenated the two best among them which were FastText (300 dimensional vectors) and Word2Vec (100 dimensional vectors) resulting in a 400 dimensional vector representation for words in the corpus.

The resulting concatenation of word embeddings yields 4 types of words: **type 1)** words which exist in both embeddings; **type 2)** words which exist in the first embedding but do not exist in the second; **type 3)** words which exist in the second embedding but do not exist in the first; **type 4)** words which neither exist in the first nor in the second embedding.

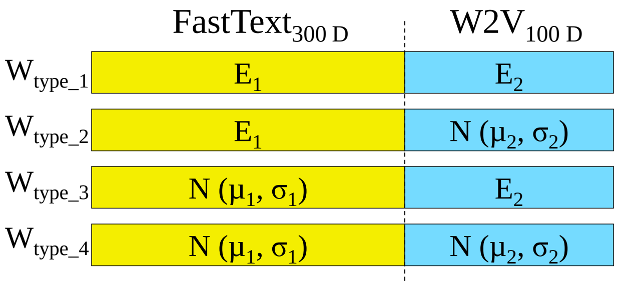

Figure 1: Assigning vectors to the respective types of words yielded from the concatenation of FastText and Word2Vec word embeddings.

The strategy adopted for assigning vectors to all four types of words is shown in Figure 1 and is explained as:
Let E_1 be vector components from the FastText embedding, E_2 be vector components from the Word2Vec embedding, μ_1 be the mean of all vectors in FastText, μ_2 be the mean of vectors in Word2Vec, σ_1 be the standard deviation of the vectors in FastText, σ_2 be the standard deviation of the vectors in Word2Vec, then the vectors assigned to the types of words are: **type 1)** get E_1 and E_2; **type 2)** get E_1 and initialize last 100 components with Gaussian distribution using μ_2 & σ_2; **type 3)** get E_2 and initialize first 300 components with Gaussian distribution using μ_1 & σ_1; **type 4)** initialize first 300 components with Gaussian distribution using μ_1, σ_1 and last 100 components with Gaussian distribution using μ_2, σ_2.

3.3. Models Used

We used four different types of neural architectures for both tasks of offensive language and hate speech detection, namely: 1) Convolutional Neural Networks (CNN); 2) Nets based on Bidirectional Long Short-Term Memory (BLSTM); 3) Nets based on Bidirectional Gated Recurrent Units (BGRU); and 4) Nets based on Bidirectional LSTMs with CNN (BLSTM+CNN). We briefly explain these architectures.

[1]http://vectors.nlpl.eu/repository/20/31.zip

[2]https://github.com/google-research/bert/blob/master/multilingual.md

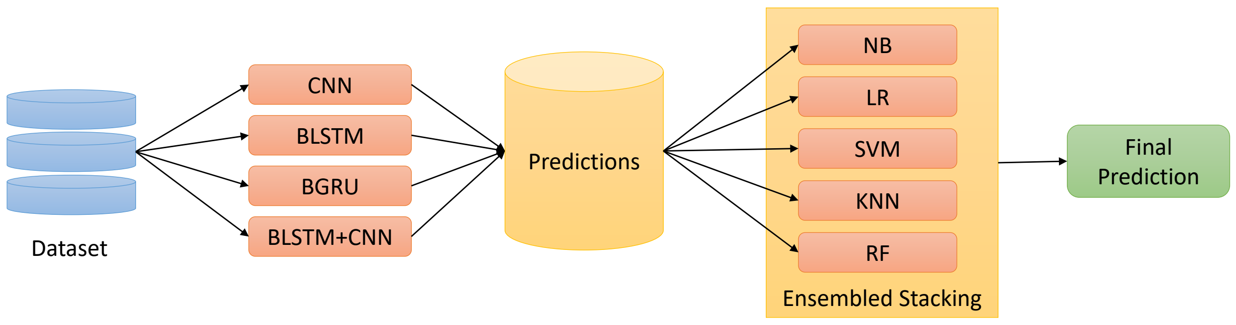

Figure 2: Classification pipeline followed to detect offensive language and hate speech in Arabic language

3.3.1. CNN

This architecture is based on the one presented by (Kim, 2014). The input layer in this architecture is an embedding layer, attached to a 1D spatial dropout layer that is then reshaped to a 2D matrix of $M \times V$, where M is maximum length of tweets in the corpus and V is the size of embedding vectors. After reshaping the input, 5 convolutional layers are attached in parallel having 128 kernels in each layer with kernel dimensions ranging from $1 \times V$, to $5 \times V$. All these parallel layers are then attached to a global max-pooling layer and concatenated to make a single feature vector, connected then to a dropout layer, followed by fully connected layers of 100, 50 and 1 units respectively. The activation function in the last layer is a sigmoid whereas for the rest of the network we use the exponential linear unit (ELU) function.

3.3.2. BLSTM

This architecture is taken from (Saeed et al., 2018). The input to this architecture is an embedding layer followed by a 1D spatial dropout layer, which is then attached to two parallel blocks of Bidirectional Long-Short Term Memory (BLSTM) where the first block has 128 units and the second block 64 units. Global max-pooling and global average-pooling layers are attached to both parallel blocks and are concatenated to make one feature vector, which is then attached to fully connected layers of 100, 50, and 1 units respectively. The activation function in the last layer is a sigmoid whereas the BLSTM layers use the tanh activation function and for the rest of the network we use the exponential linear unit (ELU) function.

3.3.3. BGRU

This architecture is also taken from (Saeed et al., 2018) and is similar to the BLSTM architecture. The only difference between this architecture and BLSTM is that we use GRU instead of LSTM. The rest of the architecture is same as that of BLSTM.

3.3.4. BLSTM+CNN

This architecture has an input embedding layer connected to a 1D spatial dropout layer. The output from the 1D spatial dropout layer is given as input to a bidirectional LSTM layer with 128 units and then a 1D convolutional layer is attached with 64 kernels of size 4, connected on its turn with a global max-pooling layer, followed by a dropout layer, and again 3 fully connected layers having 100, 50 and 1 units respectively.

3.4. Ensembled Stacking Classifier

The overall classification pipeline is shown in Figure 2. We train all four models: CNN, BLSTM, BGRU, and BLSTM+CNN, for 250, 200, 70 and 30 times respectively. The decision threshold is optimized for F1 as part of the training phase. We hence get 550 predictions for each sample in the validation set. Using these 550 predictions as a new training set, we built a stacking classifier that is an ensemble of a Naïve Bayes classifier, a Logistic Regression model, a Support Vector Machine, a Nearest Neighbours classifier and a Random Forest. We fine-tune this new Ensembled Stacking Classifier as well. We named our approach “ESOTP”, which stands for *Ensembled Stacking classifier over Optimized Thresholded Predictions of multiple deep models*.

4. Experimentation & Results

We used Keras deep learning framework with Tensorflow backend to build our deep classification pipeline. The evaluation metric used to test the classification system is macro averaged f1 score. We report cross-validation scores as our results in this paper, as there was a limit of 10 submissions at maximum per team during the OCAST4 testing phase.

4.1. Hyper-parameter Tuning

We tune hyper-parameters of the deep models used in this study by mixing grid search with manual tuning. The hyper-parameters include batch size, optimizers, learning rate, the number of kernels in CNN, the number of units in recurrent layers, and the dropout rates. The hyper-parameters in ensembled stacking classifier include penalty, solver and regularization parameter for Logistic Regression; penalty, kernel function, regularization parameter and gamma for Support Vector Machine; values of K in Nearest Neighbours; and number of estimators, splitting criterion and max. depth of trees in Random Forest.

4.2. Pre-trained Word Vectors

We compared pre-trained word embeddings with CNN architecture over 20 runs only due to time limitations. The average of 20 runs for both subtasks is shown in Table 1, which shows that Word2Vec and FastText achieved the

Word Embeddings	OFF	HS
Bert Multilingual	83.10 ± 0.75	73.45 ± 1.09
AraVec-300-SG	77.94 ± 0.28	72.64 ± 1.57
AraVec-300-CBOW	77.62 ± 0.70	72.77 ± 1.39
AraVec-100-SG	77.51 ± 0.39	72.95 ± 1.52
AraVec-100-CBOW	77.97 ± 0.53	72.56 ± 1.33
Word2Vec	**87.03** ± **0.33**	**75.98** ± **1.21**
FastText	**87.13** ± **0.23**	**76.68** ± **1.04**

Table 1: Comparison of pre-trained word embeddings averaged over 20 runs for macro f1 score.

highest f1 scores on our cross-validation when used as pre-trained word vectors, and therefore we selected both these embeddings to concatenate them for the representation of words from both embeddings.

4.3. Main Results

The main results are shown in Table 2. Naïve Bayes (NB), Logistic Regression (LR), Random Forest (RF) and Support Vector Machines (SVM) give lower F1 scores as compared to deep models in our cross-validation. Besides the deep models described in section 3.3., we trained two additional deep models: 1) BLSTM with Attention; 2) BLSTM with some statistical features like number of punctuation marks, number of characters, number of words, number of rare words, number of out-of-vocabulary words, etc. The cross-validation scores showed deterioration instead of improvement, therefore, we ignored them from being into our ensembled stacking classification.

The scores in Table 2 shown for CNN are averaged over 250 runs, for BLSTM over 200 runs, for BGRU over 70 runs and for BLSTM+CNN over 30 runs. The scores of "ESTOP" are marked with asterisk (*) sign because we split the validation set further (into train and validation) to fine-tune the ensembled stacking classifier.

Models	OFF	HS
NB+TF.IDF	64.73	48.87
LR+TF.IDF	84.55 ± 0.22	71.12 ± 0.38
RF+TF.IDF	80.92 ± 0.46	72.41 ± 1.25
SVM+TF.IDF	84.86 ± 0.29	72.88 ± 0.11
CNN	88.67 ± 0.47	75.68 ± 1.04
BLSTM	89.02 ± 0.43	76.83 ± 1.40
BGRU	88.75 ± 0.38	76.63 ± 1.36
BLSTM+CNN	87.84 ± 0.42	75.82 ± 1.42
ESTOP	95.51*	77.79*

Table 2: Macro averaged F1 cross-validation scores for both subtasks

We submitted predictions from CNN, BLSTM, BGRU, BLSTM+CNN and ESTOP for the actual test set one-by-one. The test scores indicated that CNN, BLSTM and BGRU were over-fitting whereas BLSTM+CNN was under-fitting. Overall, ESTOP approximated better generalized predictions for the actual test set as it achieved 87.37% f1 for subtask A (ranked 6/35) and 79.85% for subtask B (ranked 5/30).

5. Conclusion

We present our submission to the shared tasks of offensive language and hate speech detection in OCAST4 2020. To develop a good classification pipeline for both tasks, we select the empirically best word representations using available pre-trained word embeddings with some language-specific pre-processing, and afterwards compare a number of deep learning approaches. Our final submission is based on fine-tuning a stacking classifier where we use an ensemble of multiple models as the stacking classifier, built over different deep models trained for several times. Our classification pipeline (ESTOP) results in better generalization as compared to individual deep models.

6. Acknowledgements

We thank Louis Bruyns Foundation, Belgium, for their support in this research study.

7. Bibliographical References

Albadi, N., Kurdi, M., and Mishra, S. (2018). Are they our brothers? analysis and detection of religious hate speech in the arabic twittersphere. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 69–76. IEEE.

Chowdhury, A. G., Didolkar, A., Sawhney, R., and Shah, R. R. (2019). Arhnet - leveraging community interaction for detection of religious hate speech in arabic. In Fernando Alva-Manchego, et al., editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 2: Student Research Workshop*, pages 273–280. Association for Computational Linguistics.

Gambäck, B. and Sikdar, U. K. (2017). Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online, ALW@ACL 2017, Vancouver, BC, Canada, August 4, 2017*, pages 85–90.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*

Haidar, B., Chamoun, M., and Serhrouchni, A. (2017). Multilingual cyberbullying detection system: Detecting cyberbullying in arabic content. In *2017 1st Cyber Security in Networking Conference (CSNet)*, pages 1–8. IEEE.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Lee, S. and Kim, H. (2015). Why people post benevolent and malicious comments online. *Commun. ACM*, 58(11):74–79.

Magdy, W., Darwish, K., and Weber, I. (2015). # failedrevolutions: Using twitter to study the antecedents of isis support. *arXiv preprint arXiv:1503.02401*.

Mohaouchane, H., Mourhir, A., and Nikolov, N. S. (2019). Detecting offensive language on arabic social media using deep learning. In Mohammad A. Alsmirat et al., editors, *Sixth International Conference on Social Networks Analysis, Management and Security, SNAMS 2019, Granada, Spain, October 22-25, 2019*, pages 466–471. IEEE.

Mubarak, H., Darwish, K., and Magdy, W. (2017). Abusive language detection on arabic social media. In *Proceedings of the First Workshop on Abusive Language Online*, pages 52–56.

Mubarak, H., Darwish, K., Magdy, W., Elsayed, T., and Al-Khalifa, H. (2020). Overview of osact4 arabic offensive language detection shared task. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools (OSACT)*, volume 4.

Saeed, H. H., Shahzad, K., and Kamiran, F. (2018). Overlapping toxic sentiment classification using deep neural architectures. In *2018 IEEE International Conference on Data Mining Workshops, ICDM Workshops, Singapore, Singapore, November 17-20, 2018*, pages 1361–1366. IEEE.

Santosh, T. Y. S. S. and Aravind, K. V. S. (2019). Hate speech detection in hindi-english code-mixed social media text. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, COMAD/CODS 2019, Kolkata, India, January 3-5, 2019*, pages 310–313.

Soliman, A. B., Eissa, K., and El-Beltagy, S. R. (2017). Aravec: A set of arabic word embedding models for use in arabic NLP. In *Third International Conference On Arabic Computational Linguistics, ACLING 2017, November 5-6, 2017, Dubai, United Arab Emirates*, pages 256–265.

Warner, W. and Hirschberg, J. (2012). Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, Montréal, Canada, June. Association for Computational Linguistics.

Waseem, Z. and Hovy, D. (2016). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Yin, D., Xue, Z., Hong, L., Davison, B. D., Kontostathis, A., and Edwards, L. (2009). Detection of harassment on web 2.0. *Proceedings of the Content Analysis in the WEB*, 2:1–7.