Quick and Simple Approach for Detecting Hate Speech in Arabic Tweets

Abeer Abuzayed
Islamic University of Gaza
Gaza, Palestine
aabuzayed1@students.iugaza.edu.ps

Tamer Elsayed
Qatar University
Doha, Qatar
telsayed@qu.edu.qa

Abstract

As the use of social media platforms increases extensively to freely communicate and share opinions, hate speech becomes an outstanding problem that requires urgent attention. This paper focuses on the problem of detecting hate speech in Arabic tweets. To tackle the problem efficiently, we adopt a "quick and simple" approach by which we investigate the effectiveness of 15 classical (e.g., SVM) and neural (e.g., CNN) learning models, while exploring two different term representations. Our experiments on 8k labelled dataset show that the best neural learning models outperform the classical ones, while distributed term representation is more effective than statistical bag-of-words representation. Overall, our best classifier (that combines both CNN and RNN in a joint architecture) achieved 0.73 macro-F1 score on the dev set, which significantly outperforms the majority-class baseline that achieves 0.49, proving the effectiveness of our "quick and simple" approach.

Keywords: Offensive language, Twitter, Text classification, Learning models, Neural models, Distributed term representation.

1. Introduction

Twitter is a place where 330 million users (in 2019)[1] from every background, race, religion, and nationality interact and communicate, and freely share their ideas, opinions, and beliefs. This makes Twitter easy to exploit in sharing content that targets and threatens individuals or groups based on their common characteristics or identities by spreading hate speech. According to Twitter hateful conduct policy[2], hate speech is to "attack or threaten other people on the basis of race, ethnicity, national origin, caste, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease", such as the tweet: "يا خوارج يا ارهابيين يا منبع اليهود" (O Kharijites, terrorists, the source of the Jews). Twitter encourages users to report any kind of hate speech that violates the hateful conduct policy, so that an action can be made such as suspending the user or deleting the tweet. Despite the considerable effort that social media sites are making in trying to curb hate speech, it is still threatening the online communities and users are still seeing it on many platforms. As hate speech might result in serious physical or mental abuse, there is an imperative need to detect and prevent such content on social media platforms.

Several researchers studied hate speech in the social media domain and proposed various approaches to detect it with more focus on English language, e.g., Malmasi and Zampieri (2018), Watanabe et al. (2018), Zhang and Luo (2018), and Zhang et al. (2018). However, detecting hate speech in Arabic content is still nascent. The richness and complexity of the nature and structure of the Arabic language, the variety of dialects, and the problems at orthographic, morphological, and syntactic levels make detecting hate speech in Arabic very challenging.

In this work, we conduct a preliminary study on the detection of hate speech in Arabic tweets as part of our participation in the Hate Speech Detection subtask in OSACT4 workshop[3] (Mubarak et al., 2020). Given the tight time we had for participation[4], we aim to tackle the classification problem in a simple, quick, yet effective approach. We elect to use "simple" features that are not problem-specific but easy to compute or use, while leveraging the richness, maturity, and strong support for "quick" development that current popular machine learning frameworks (e.g., Keras) provide. Adopting this *quick* and *simple* approach for developing our classification system for hate speech detection, we investigate the performance of several learning models and aim to answer two research questions in the context of this problem:

RQ1. Is distributed (latent) word representation (e.g., Word2Vec embeddings) more effective than standard statistical bag-of-words representation (e.g., tf-idf)?

RQ2. Are neural models more effective than classical machine learning models?

To answer both questions, we conducted experiments over seven classical and eight neural learning models using the labelled dataset of 8,000 tweets, provided by the shared task organizers, and submitted two runs on the test set. Our results show that, surprisingly, the bag-of-words tf-idf representation is more effective than distributed word embeddings representation; however the best neural models outperform classical models. Overall, our best classifier achieved a reasonable 0.73 macro-F1 score on the dev set, which significantly outperforms the majority-class baseline that achieves 0.49, proving the effectiveness of our quick and simple approach.

Our contribution in this work is two-fold:

1. We conducted a preliminary study investigating the performance of 15 different classical and neural learning models for detecting hate speech in Arabic tweets.
2. We demonstrated a simple and quick approach of developing a system that is implemented in less than 3 days to tackle the problem, yet achieved reasonable performance. We make all of our code open-source for the research community[5].

The paper is organized as follows. Section 2 describes related work. Section 3 outlines our approach in tackling the problem. Section 4 presents our experimental

1 https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

2 https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy

3 http://edinburghnlp.inf.ed.ac.uk/workshops/OSACT4/

4 We only had 3 days before the submission deadline.

5 https://github.com/AbeerAbuZayed/QUIUG_Hate-Speech-Detection_OSACT4-Workshop

evaluation results. Section 5 concludes our work with potential future work.

2. Related Work

As mentioned earlier, there are several research studies conducted to study hate speech in online communities over English content. Mondal et al. (2017) conducted a study in online social media to understand how social media platforms are rich with hate speech and to investigate the most popular hate expressions and the main targets of online hate speech. Malmasi and Zampieri (2018) aimed to distinguish hate speech from general profanity using a dataset annotated as "hate, offensive, and ok", with advanced ensemble classifiers and stacked generalization along with various features such as n-grams, skip-grams, and clustering-based word representations. Additionally, Watanabe et al. (2018) classify tweets based on three labels (clean, offensive and hateful) using sentiment-based features, semantic features, unigram features, and pattern features. Zhang and Luo (2018) and Zhang et al. (2018) also conducted studies on Twitter hate speech for the English language.

Other researchers focused on detecting *offensive language* over *Arabic* content, where a number of studies were conducted to detect offensive and abusive language for Arabic Tweets and for YouTube comments (Mubarak and Darwish, 2019; Alakrot et al., 2018; Mohaouchane et al., 2018; Mubarak et al., 2017). However, *hate speech* is different from offensive and abusive language (Malmasi and Zampieri, 2018). Also, Zhang and Luo (2018) argue the same point and pointed out that the term "hate speech" might be overlapping with other terms such as "offensive", "profane" and "abusive". In order to distinguish them, they defined hate speech as "targeting individuals or groups on the basis of their characteristics and demonstrating a clear intention to incite harm, or to promote hatred and this speech may or may not use offensive or profane words".

Consequently, hate speech should be distinguished from other offensive and profane languages. Thus, other studies focus only on hate speech detection. Albadi et al. (2018) developed a system to detect religious hate speech in Arabic tweets. They used three various approaches to tackle this problem. Firstly, they constructed an Arabic lexicon of religious hate speech and used it to classify tweets to "hate" if the tweet terms exist in the lexicon, otherwise it is labelled as "not hate". Secondly, they trained Logistic Regression and SVM classifiers using n-gram models. Finally, a GRU model with the pre-trained embedding model AraVec (Twitter-CBOW 300D architecture) showing 0.77 F1 score was adopted.

Moreover, Chowdhury et al. (2019) studied religious hate speech in Arabic tweets too, where they argued that considering the community interactions can raise the ability to detect hate speech content on social media. To investigate this, Arabic word embedding (AraVec, Twitter-CBOW 300D architecture), social network graphs, and neural networks (e.g., RNN+CNN) were used. They pointed out that considering community interactions significantly improves the result and outperforms Albadi et al. (2018) performance, where the combination of social network graphs and joint LSTM and CNN model achieved 0.78 F1 score.

Furthermore, there are studies on hate speech detection in multilingual tweets including Arabic. Ousidhoum et al. (2019) used the bag of words (BOW) as features with Logistic Regression (LR) and deep learning models to detect hate speech in multilingual tweets. Smedt et al. (2018) conducted an experiment to detect online Jihadist hate speech in multilingual tweets, where SVM was used to classify tweets.

In this study, we focus on detecting hate speech in Arabic tweets using several classical and neural learning models with tf-idf and word embeddings features. We adopt a quick and simple approach of developing our classifiers and conducting our experiments, focusing on unigram representations that are problem-independent, while leveraging the power and ease-of-use of existing learning frameworks.

3. Approach

We approach hate speech detection as a supervised learning problem. In our study, we experimented with several classical and neural learning models trained for detecting Arabic hate speech on Twitter. We adopted basic text preprocessing and two main feature extraction techniques for comparison.

3.1 Preprocessing

To prepare our dataset for the feature extraction process, basic text preprocessing is done as follows:

- Punctuations, foreign characters and numbers (including user mentions and URLs), and diacritics (tashdid, fatha, tanwin fath, damma, tanwin damm, kasra, tanwin kasr, sukun, and tatwil/kashida) are all removed. We also removed repeated characters.
- The remaining Arabic text is normalized. Letters are normalized as follows:
  - {"إأآا" to "ا"}
  - {"ي" to "ى"}
  - {"ؤ" to "ء"}
  - {"ئ" to "ء"}
  - {"ة" to "ه"}
  - {"گ" to "ك "}

While some normalization has been done through building the pre-trained word embedding model (AraVec2.0) used in our experiment, we augmented it with additional steps.

3.2 Feature Extraction

We adopted two main simple and problem-independent feature extraction techniques: tf-idf and word embeddings.

Firstly, tf-idf term weight (term frequency-inverse document frequency) indicates how relevant a term is to a document in a collection of documents. In our experiments, tf-idf weights are only used with the classical machine learning algorithms in order to compare against using word embeddings as features.

Secondly, word embeddings are the most popular distributed representation of words (or terms). Each word in the vocabulary is represented as a vector of a few hundred dimensions, where words that have the same meaning are closer to each other, while the words with different meanings are far apart. This is done by learning the vector representation of the words through the contexts in which they appear. One of the popular techniques for efficiently learning a standalone word embedding from a text corpus is Word2Vec (Mikolov et al., 2013). There are two different learning models to learn the embeddings, Skip Gram and Continuous Bag of Words (CBOW). The CBOW model learns the embeddings by predicting the current word using the context as an input, while the continuous skip-gram takes the current word as input and learns the embeddings by predicting the surrounding words (Mikolov et al., 2013).

In our experiments, we used the pre-trained Arabic word embedding model AraVec2.0 (Soliman et al., 2017), which provides various pre-trained Arabic word embedding model architectures; each is trained on one of three different datasets: tweets, Web pages, and Wikipedia Arabic articles. Moreover, for each dataset, two models are built: one using Skip Gram and another using CBOW. For the purpose of this study, we used the pre-trained SkipGram 300D-embeddings trained on more than 77M tweets, since we work on tweets. We used the pre-trained model in both classical and neural learning approaches. To use it with classical learning algorithms, the average vector of all the embeddings of the tweet words is computed and used as the feature vector of the tweet. However, for the neural learning models, the embedding vectors are used to initialize the weights of the embedding layer, which is then connected to the rest of the layers in the network.

3.3 Models

This section describes the classical and neural learning models used in our experiments.

3.3.1 Classical Learning Models

We experimented with various classical machine learning models, namely SVM, Random Forest, XGBoost, Extra Trees, Decision Trees, Gradient Boosting, and Logistic Regression. These models are trained along with both types of features we described earlier, tf-idf and pre-trained word embeddings.

3.3.2 Neural Learning Models

We experimented with two types of neural models, Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN). We tried different RNN architectures, namely Long Short-Term Memory (LSTM), Bidirectional LSTM (BLSTM), and Gated Recurrent Unit (GRU).

We also tried a combination of both CNN and RNN. Previous studies showed that the joint CNN and RNN architecture outperforms CNN or RNN alone in natural language processing tasks such as sentiment analysis and text classification tasks (Wang et al., 2016 and Zhou et al. 2015). This combined architecture allows the network to learn local features from the CNN, and long-term dependencies, positional relation of features, and global features from the RNN (Wang et al., 2016). The combined architecture used in this work consists of one CNN layer with max-pooling and time distributed layer, followed by one RNN layer and dropout layer, as shown in the example joint CNN and LSTM architecture in Figure 1.

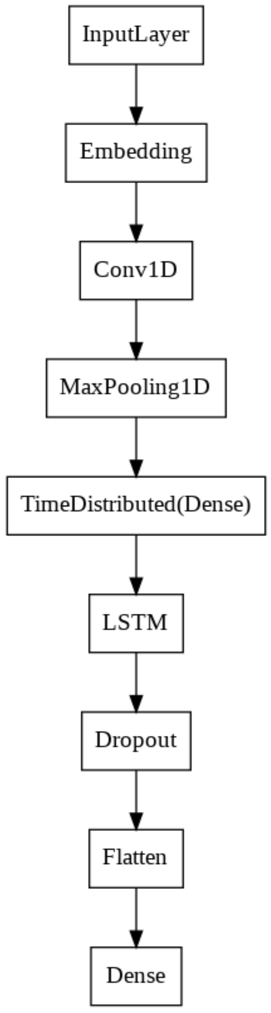

Figure 1: Joint CNN and LSTM model architecture.

4. Experimental Evaluation

In this section, we present and analyze the performance of our trained models. We start with the experimental setup, followed by the analysis of the two experiments we conducted to answer the two research questions. Finally, we discuss the results of our two submitted runs to the shared task.

4.1 Experimental Setup

For the purpose of this study, we use SemEval 2020 Arabic offensive language dataset (OffensEval 2020, Subtask B for detecting hate speech) (Mubarak et al., 2020). The dataset was split into train, dev, and test sets (70%, 10%, and 20% respectively). There are 7,000 training tweets, only 361 of them (about 5.2%) are labelled as hate speech. There are 1,000 dev tweets, only 44 of them (4.4%) are labelled as hate speech. This shows how the two classes in the dataset are clearly unbalanced.

As expected, we used the training set to learn each model's parameters and the dev set to tune its hyperparameters. The hyperparameters and their tuned values are listed in Table 1.

Hyperparameter	Value
Number of filters (CNN)	25
Kernel size (CNN)	5
Number of hidden units (RNN)	16
Dropout rate (Regularizer)	0.5
Learning rate (Adam optimizer)	0.001

Table 1: Tuned values of the hyperparameters.

To answer the two research questions we listed in Section 1, we conducted two main experiments. The first compares the use of tf-idf vs word embeddings features, conducted on classical machine learning models. The second compares classical vs. neural models.

We evaluated the performance of our models using two measures: macro-averaged F1 (the official shared task measure) and F1 score on the hate speech (HS) class (since the target HS class is scarce). All reported results in this section are on the dev set unless otherwise mentioned. Notice that the majority-class baseline on the dev set yields a 0.49 macro-F1 score.

It is worth noting that all of our development and experiments were performed through Google Colaboratory using Python and Keras libraries.

4.2 RQ1: tf-idf vs. Word Embeddings

To answer RQ1, we conducted an experiment over the seven classical models listed in Section 3.3.1 using both tf-idf and pre-trained word embeddings (AraVec 2.0).

Figures 2 depicts the performance of the models in each of the two cases measured in macro-averaged F1. There are several interesting observations. First, we notice that the performance using tf-idf varies from 0.49 to 0.68, while using word embeddings it varies from 0.51 to 0.57. Second, some models (e.g., SVM) exhibited slightly better performance using word embeddings, however more models (e.g., Random Forest) exhibited much better performance using tf-idf. Overall, the best three models (namely Extra Trees, Random Forest, and Gradient Boosting, respectively) are all indeed using tf-idf. This is a surprising result, since tf-idf features neither capture meaning nor are contextualized; both attributes are (or at least should be) captured by word embeddings. This observation definitely needs more investigation.

Figure 3 illustrates the performance of the models in each of the two cases, but this time measured in F1 over the positive class. It indicates very similar, but even stronger, observations. Moreover, it clearly shows that the task of detecting the HS tweets is, not surprisingly, much harder than non-HS, achieving an F1 score of 0.39 at best.

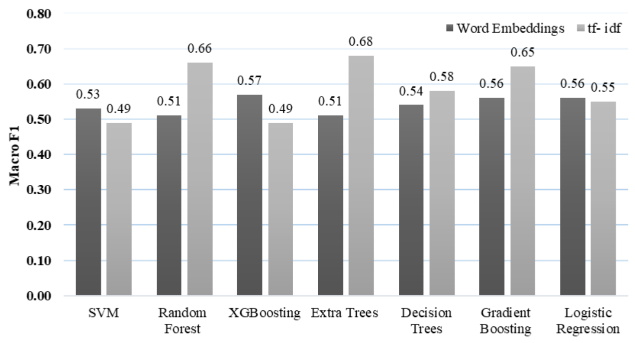

Figure 2: Macro F1 of classical learning models.

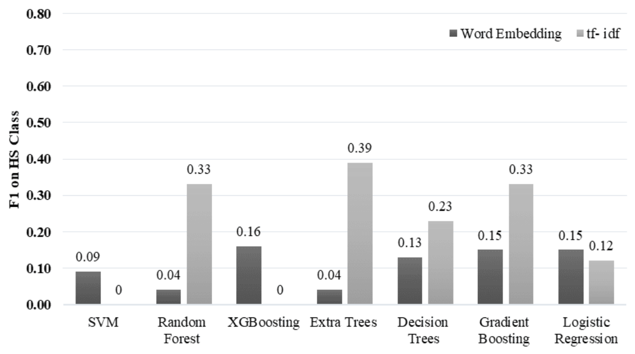

Figure 3: F1 on HS class of classical learning models.

4.3 RQ2: Classical vs. Neural Models

We now turn our attention to RQ2, which is concerned with comparing classical and neural models. We considered the best-performing classical model, i.e., Extra Trees with tf-idf features, as the *baseline*, which we compare against eight neural models:

- The first three are RNN models, namely LSTM, BLSTM, and GRU.
- The fourth is CNN.
- The next three are combined CNN and RNN models, one for each RNN type.
- The last one is a combined CNN and LSTM version that is trained on an oversampled training data to address the unbalanced data problem, where some HS (i.e., the minority class) examples are replicated.

Due to time constraints, we only trained the neural models using word embeddings. According to the results of the first experiment in Section 4.2, using tf-idf features is worth trying too. We defer this to future exploration.

Tweet	True label	Predicted (Extra Trees, Embeddings)	Predicted (Extra Trees, tf- idf)	Predicted (CNN+LSTM)
إحنا أتباع إيران يا آل سلول يا نسل اليهود We are followers of Iran, O family of Salul, descendants of the Jews.	HS	**HS**	**HS**	**HS**
بس يا فاشل يا خاين يا عميل Shut up, O loser and traitor!	NOT_HS	**NOT_HS**	**NOT_HS**	**NOT_HS**
يا هنيدي يا عمرو ياديب الاهلي ف الصداااااارة خبوا الحصالة اددددداااا 😂😂 #للخلف_درر_يا_زمالك O Hinaidi, O Amr Adeeb, Al-Ahly is taking the lead. #GoBack_Zamalek team.	HS	NOT_HS	**HS**	NOT_HS
يا كافر يا زنديق يا مرتد يا انت عاوز يبقى عندنا ديمقراطية زى الكفرة اللى ما يعرفوش ربنا O bastard and Godless! You want us to have a democracy like the infidels, who do not believe in God.	NOT_HS	**NOT_HS**	**NOT_HS**	HS
ههههههههههههههههههههههه يا طحلبي يا صغير جدة يا جاهل شوف بطولات الاتحاد قبل 1417 Hahahahahahaha! You, little kid of Jeddah, you ignorant. Check the Al-Ittihad [tournaments/ championships] before 1417.	HS	NOT_HS	NOT_HS	**HS**

Table 2: Examples (from the dev-set) of correct (bolded) and incorrect (underlined) classification using Extra Trees models with word embeddings and tf- idf and the combined CNN+LSTM neural model.

Figure 4 depicts the performance of all tried neural models along with the baseline measured in macro-averaged F1. Similar to the first experiment, we have several interesting observations. First, the figure shows that combining CNN and RNN models improved performance over individual models. Second, the classical model unexpectedly exhibits a comparable performance to several neural models. Third, combining CNN and LSTM exhibited the best performance, outperforming all other neural models in addition to the baseline classical model. Finally, oversampling did not help, at least when applied to the best performing model.

Figure 5 indicates the same exact performance patterns measured in F1 on the HS class. However, the performance gap between the best neural model and the baseline is even widened.

Table 2 shows examples (from the dev-set) of correct and incorrect classification using both of tf-idf and word embeddings with Extra Trees classifier and combined CNN and LSTM neural model. The table shows that the models made different mistakes.

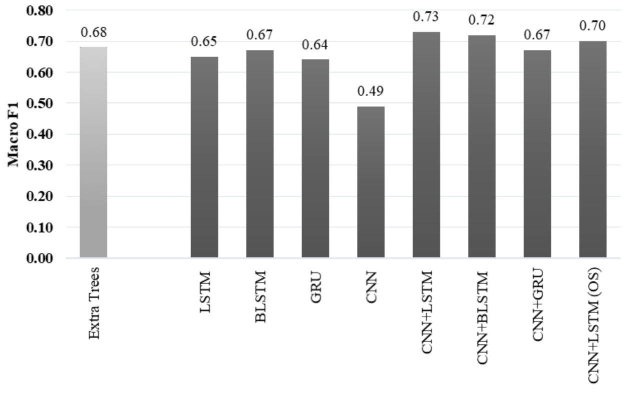

Figure 4: Macro F1 of neural learning models compared to the best-performing classical model.

4.4 Submitted Runs

Based on the results above, we chose the combined CNN and LSTM model in addition to its oversampling version to submit results on the test set to the shared task. Table 3 shows the results of the two models as reported by the task organizers, compared to the results on the dev set. As expected, the performance on the test set is slightly lower than on the dev set, however the unsampled version still outperforms the oversampled one.

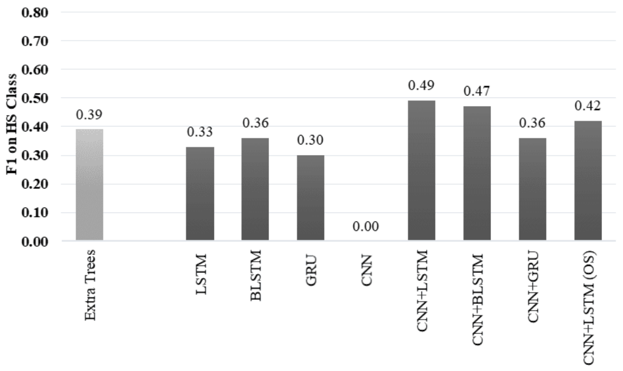

Figure 5: F1 on HS class of neural learning models compared to the best-performing classical model.

Model	Macro F1 (dev-set)	Macro F1 (test-set)
CNN+LSTM	**0.73**	**0.69**
CNN-LSTM (OS)	0.70	0.65

Table 3: Macro F1 scores of submitted models.

5. Conclusion and Future Work

In this paper, we presented a quick and simple approach to tackle the problem of detecting hate speech in Arabic tweets. Our approach adopts simple problem-independent features to represent terms in tweets and leverages the quick development service supported by existing powerful machine learning libraries. We compared 15 classical and neural learning models along with two different term representations (tf-idf and word embeddings). Our experiments over 8k labelled dataset of Arabic tweets showed that tf-idf representation is more effective than word embeddings when used in classical models, and that the best neural learning model (a joint CNN and LSTM architecture) outperforms the classical ones. To our knowledge, this is the first time a combined CNN and LSTM is used to detect hate speech over Arabic tweets. The classification performance achieved by this combined model exhibited a significant improvement over the majority-class baseline, proving the effectiveness of our "quick and simple" approach.

For future work, we plan to conduct several experiments. Firstly, as it shows better performance with classical learning models, we will consider using tf-idf representation with neural models as well. Secondly, we plan to experiment with transfer learning techniques to leverage the models that are trained for related tasks such as offensive language detection. Thirdly, we will further investigate the sampling techniques to overcome the unbalanced data problem. Finally, since the pre-trained model BERT yields the state of the art performance in several natural language processing tasks (Devlin et al., 2018), it is worth trying for hate speech detection too.

6. References

Alakrot, A., Murray, L. and Nikolov, N.S., 2018. Towards accurate detection of offensive language in online communication in arabic. *Procedia computer science*, 142, pp.315-320.

Albadi, N., Kurdi, M. and Mishra, S., 2018, August. Are they our brothers? Analysis and detection of religious hate speech in the Arabic Twittersphere. In 2018 *IEEE/ACM ASONAM* (pp. 69-76).

Chowdhury, A.G., Didolkar, A., Sawhney, R. and Shah, R., 2019. ARHNet-Leveraging Community Interaction for Detection of Religious Hate Speech in Arabic. In *Proceedings of ACL: Student Research Workshop* (pp. 273-280).

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Malmasi, S. & Zampieri, M., 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence,* 30(2), pp. 187-202.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. *In Advances in neural information processing systems* (pp. 3111-3119).

Mohaouchane, H., Mourhir, A. and Nikolov, N.S., 2019, October. Detecting Offensive Language on Arabic Social Media Using Deep Learning. In *IEEE SNAMS* (pp. 466-471).

Mondal, M., Silva, L. A. & Benevenuto, F., 2017. *A Measurement Study of Hate Speech in Social Media..* New York, USA, ACM, p. 85–94.

Mubarak, H. & Darwish, K., 2019. Arabic Offensive Language Classification on Twitter. *In International Conference on Social Informatics* (pp. 269-276). Springer.

Mubarak, H., Darwish, K. and Magdy, W., 2017, August. Abusive language detection on Arabic social media. *In Proceedings of the First Workshop on Abusive Language Online* (pp. 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)*, vol. 4.

Ousidhoum, N., Lin, Z., Zhang, H., Song, Y. and Yeung, D.Y., 2019. Multilingual and Multi-Aspect Hate Speech Analysis. *arXiv preprint arXiv:1908.11049.*

Smedt, T., De Pauw, G. and Van Ostaeyen, P., 2018. Automatic detection of online jihadist hate speech. *arXiv preprint arXiv:1803.04596.*

Soliman, A.B., Eissa, K. and El-Beltagy, S.R., 2017. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science,* 117, pp.256-265.

Wang, X., Jiang, W. and Luo, Z., 2016. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016: Technical papers* (pp. 2428-2437).

Watanabe, H., Bouazizi, M. & Ohtsuki, T., 2018. Hate speech on Twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE Access,* 6, pp. 13825-13835.

Zhang, Z. & Luo, L., 2018. Hate speech detection: A solved problem? the challenging case of long tail on Twitter. CoRR abs/1803.03662 (2018).

Zhang, Z., Robinson, D. & Tepper, a. J., 2018. Hate Speech Detection Using a Convolution-LSTM Based Deep Neural Network. *In Proceedings of ACM WWW conference (WWW'2018).* New York, NY, USA.

Zhou, C., Sun, C., Liu, Z. and Lau, F., 2015. A C-LSTM neural network for text classification. arXiv preprint arXiv:1511.08630.