Cyberbullying over Lebanese Social Media - EECE 463 Milestone IV

Ali Yassine – Bahaa Kaaki

I. Description of Dataset

The dataset used is "Arabic Levantine Hate Speech Detection" dataset obtained from https://www.kaggle.com/haithemhermessi/arabic-levantine-hate-speech-detection. This dataset consists of 5789 unique values that implements multi-class classification: Normal, Abusive and Hate. Normal tweets with no offensive, aggressive or insulting content, Abusive tweets, which combine offensive, aggressive, and insulting content, and Hate tweets, which contain or dedicate abusive language towards a person, or demeans a group of people based on their identity. The size of the dataset is around 706 KB. The dataset is unbalanced in regard to the distribution of tweets between normal, abusive and hate, as it is made up of 62% normal, 30% abusive and 8% hate. To obtain an accurate and reliably trained model, we worked on balancing the dataset, which we converted to having 34% normal, 33% abusive and 33% hate. In both datasets the split was 70% training, 20% cross validation, and 20% testing.

II. Implementation of ML solution

In our project we used supervised learning for hate speech detection. To approach this, we used various classical and neural learning models that are trained for detecting hate speech on Twitter in the Arabic language. To do this we started with text preprocessing and feature extraction.

A. Data preprocesssing

Prior to the feature extraction stage, we had to perform some text preprocessing to remove certain characters and features. We removed punctuations, numbers, URLs, foreign characters such as "@", letters that are consecutively repeated, and diacritics which are common in Arabic texts. Some letters were also replaced as shown below:

- "گ" normalized to "ك"
- "ؤ" normalized to "ء"
- "ئ" normalized to "ء"
- "ي" normalized to "ى"
- "ة" normalized to "ه"
- "إأآا" normalized to "ا"

B. Feature Extraction

We used two different techniques for feature extraction:

- Tf-idf (term frequency inverse document frequency): The weight of this term shows how important a word is to a document. The value of the tf-idf increases according to the number of times a word appears in a document.

- Word Embedding: This technique is a representation for text where words that have similar meanings are placed close to each other in a vector of a few hundred dimensions and words which do not have similar meanings are placed apart. We used a pre-trained Arabic word embedding model called AraVec2.0(Soliman et al., 2017) [1]. This model contains several different architectures that support tweets and Arabic Wikipedia pages. Two models are built for each one of the datasets, one Skip Gram model and one CBOW model. Our datasets contained tweets mostly, so we decided to use a pre-trained SkipGram 300D-embeddings which were trained on around 77 million tweets. We used this model in classical and neural learning algorithms. With the classical learning algorithm, we averaged the embeddings of the different words in the tweets and used it as the feature vector of the tweet. With the neural learning algorithm, we used the embedding vectors to compute the weights of the embedding layer which is connected to the entirety of the neural network.

C. Machine Learning models

This section discusses the different models used in our experiment. The models we chose for our experiment are the classical and neural learning models.

Classical learning model:

We tested a few different classical learning models which are:

- SVM
- Random Forest
- XGBoost
- Extra Trees
- Decision Trees
- Gradient Boosting
- Logistic Regression.

We used these models with both types of features we mentioned in the previous section.

On the other hand, we used two different types of neural models:

- Recurrent Neural Network
    - Long Short-Term Memory (LSTM)
    - Bidirectional LSTM (BLSTM)
    - Gated Recurrent Unit (GRU)
- Convolutional Neural Network

We also experimented with a mixture of convolutional neural networks and recurrent neural networks because different experiments should that this setup outperforms the individual models in natural language processing task (Wang et al., 2016 and Zhou et al. 2015) [2] [3]. The combination allowed the model to learn local features from the conventional neural network and global features, long term dependencies and positional relation of features from the recurrent neural network. The combination used consists of one convolutional neural network layer with time distributed layer and max pooling followed by a recurrent neural network layer and a dropout layer. For our implementation, we relied on 3 major libraries: Keras for the implementation of the different model and their architectures, sklearn for the preprocessing and the

performance measures, and matplotlib for the plotting of the curves.

III. Testing Results

A. *Initial Step – Unbalanced dataset*

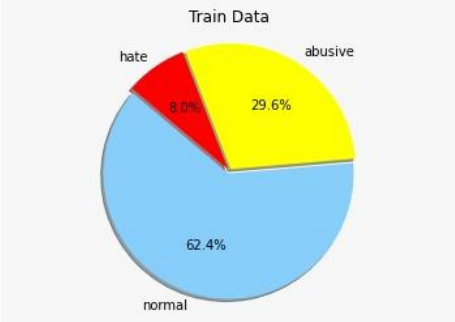


Chart showing the percentage of distribution of tweets in the unbalanced training dataset.

The initial training dataset that our models were going to be trained on was an unbalanced dataset. This dataset had tweets distributed in three different classes: normal, abusive, and hate. However, the number of tweets present in each of the three distinct classes was different, and the distribution was not equally made. Around 62.4% of the tweets available in the dataset is classified as *normal*, 29.6% is classified as *abusive*, and 8.0% is classified as *hate*.

B. *Improving upon initial step – Balancing the dataset*

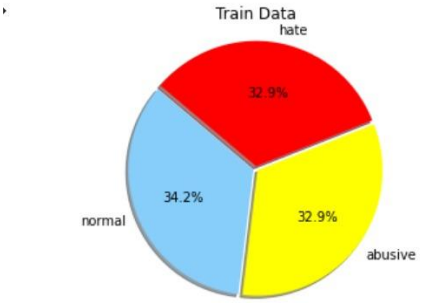
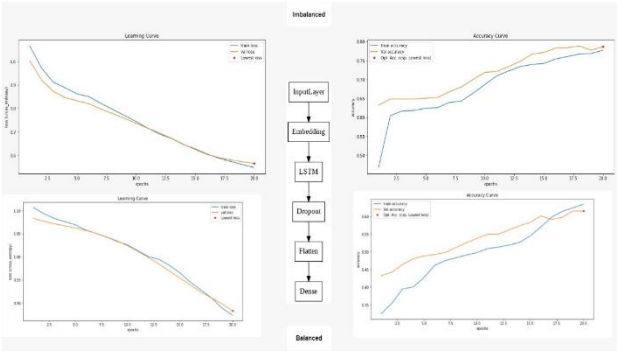


Chart showing the percentage of distribution of tweets in the balanced training dataset.

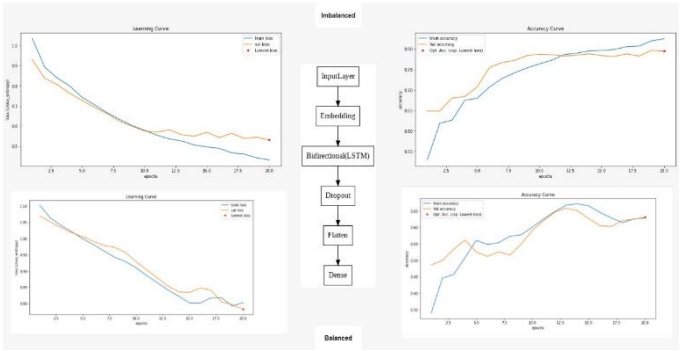
Since having an unbalanced training dataset would result in bias in the testing phase, balancing the training set is a crucial step to take. Hence, the dataset was balanced in a way to make all three classes contain around the same number of tweets. Approximately 34.2% of the tweets available in the balanced dataset is classified as *normal*, 32.9% is classified as *abusive*, and 32.9% is classified as *hate*. This balancing step will assure that our models will give better results that are less biased to the dataset. It will also allow our models to be able to generalize more, and therefore perform better on the testing set.

C. *Learning and Accuracy curves- Balanced and Unbalanced*

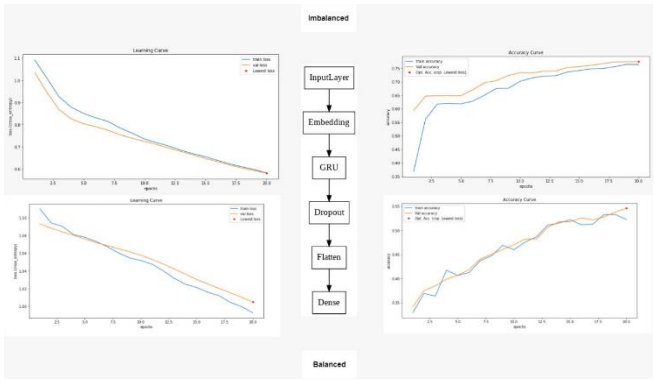
Below, we are showing the learning and accuracy curves of the balanced and imbalanced datasets for all the NN models. All the training was done with 20 epochs and a batch size of 2000, a quick grid search on these two hyperparameters from a range of different epoch numbers, and batch sizes, showed us that the best performance on these 2. The higher number of batch size helped decrease the divergence in the loss curve that we were encountering in some scenarios at the end of the training. All the NN model architectures were selected.



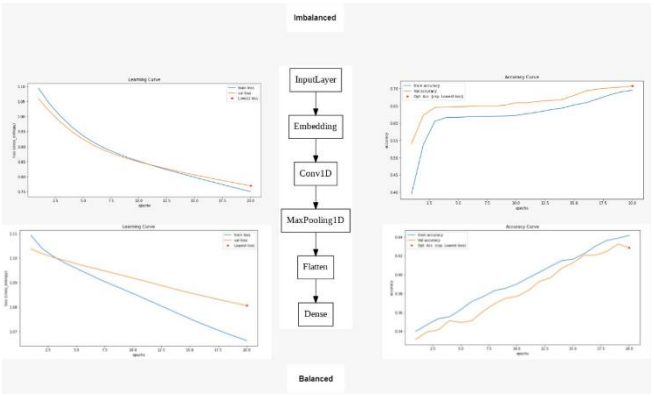
Learning and Accuracy curves for the Balanced and Unbalanced datasets for the LSTM model.



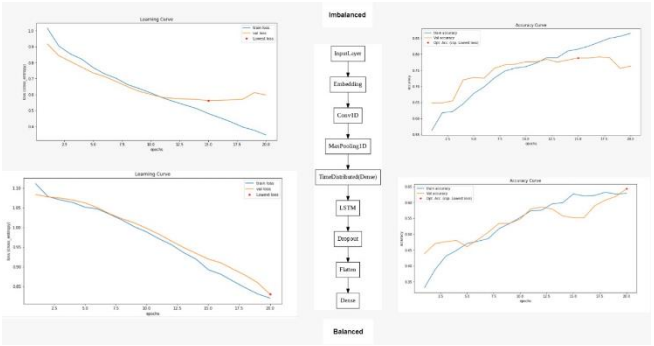
Learning and Accuracy curves for the Balanced and Unbalanced datasets for the Bidirectional LSTM model.



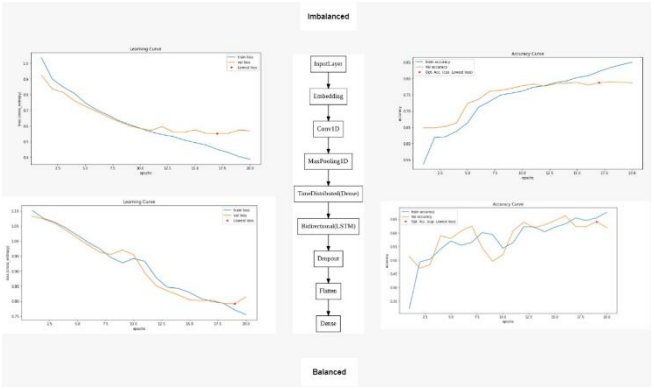
Learning and Accuracy curves for the Balanced and Unbalanced datasets for the GRU model.



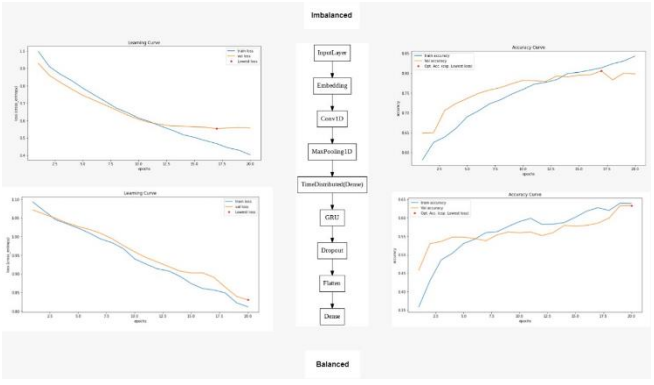
Learning and Accuracy curves for the Balanced and Unbalanced datasets for the Conv1D model.



Learning and Accuracy curves for the Balanced and Unbalanced datasets for the Conv1D and LSTM combined model.



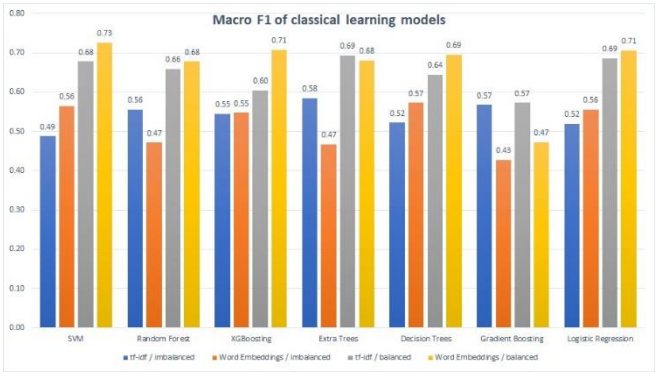
Learning and Accuracy curves for the Balanced and Unbalanced datasets for the Conv1D and Bidirectional LSTM combined model.



Learning and Accuracy curves for the Balanced and Unbalanced datasets for the Conv1D and GRU combined model.

D. Results on classical models- Balanced dataset

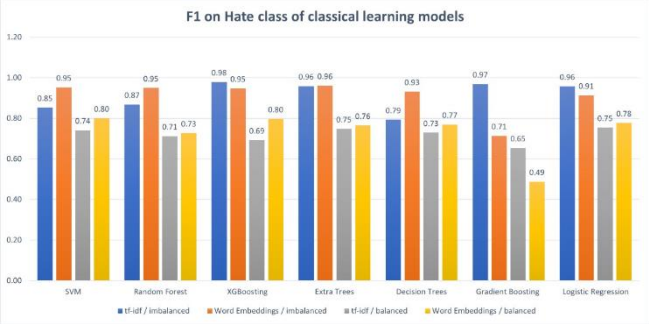
In order to properly compare the performances of our different models and our two datasets, we had to select a criterion for which we compare on. Several factors had to be taken into consideration when it came to the selection. The accuracy was not a great indicator especially for the unbalanced dataset as it is biased towards, and favors the classes with more entries, which is the Normal class in this case. Precision is an important measure and especially for the abusive and hate classes, as if the precision is too low, a lot of people will start getting normal tweets classified as abusive or hate tweets, which can cause them to get undeserved bans and punishments and might lead to users abandoning the site entirely. Similarly, Recall is also a very important measure as a low recall indicates that the model is not correctly classifying the tweet in a lot of cases, and so a low recall in the abusive or hate classes would mean that a lot of abusive or hateful tweets are going unnoticed which goes against our entire project. Thus, it goes without saying that an aggregator between both precision and recall would be a good indicator of the model's performance. This is why we selected the F1 score as our criteria for comparison. The F1 score considers both the precision and recall of the model and works well on both even and uneven class distributions. Initially, we simply took the macro average F1 score of the classical learning models and compared them. We can clearly notice the significant improvement in the performance of the models on the balanced dataset. The addition of word embedding instead of the TF-IDF technique also produced an improvement in the macro average F1 score. Overall, out of all the classical models we tested on, the best preforming model was the SVM. The SVM produced a macro average F1 score of 0.73 on the balanced dataset using the word embeddings. In contrast, the Gradient Boosting model preformed badly across all 4 tests, seemingly not working well with the word embedding.



Histogram showing the Macro F1-score for all the classical learning models and for each of the tf-idf for unbalanced dataset, the tf-idf for balanced dataset, word embeddings for unbalanced dataset, and word embeddings for balanced dataset.

As our model concentrates more on correctly identifying hate and abusive tweets, we wanted to record the individual class F1 score for both the *Hate* class and the *Abusive* class. The results for the *hate* class were good across the board. The SVM and DT models were the most consistent. The gradient boost performed badly on the balanced dataset particularly. Overall, the imbalanced dataset produced slightly biased results due to the very low number of hate entries in the imbalanced dataset. This was causing the recall of the hate class to be extremely high, which was skewing the F1 score a bit. The fact that the abusive class was the intermediary class in between the two extremes of normal and hate, and due to the very apparent similarities between the abusive and hate tweets, a lot of the tweets that should have been classified as abusive were incorrectly classified as hate as well.

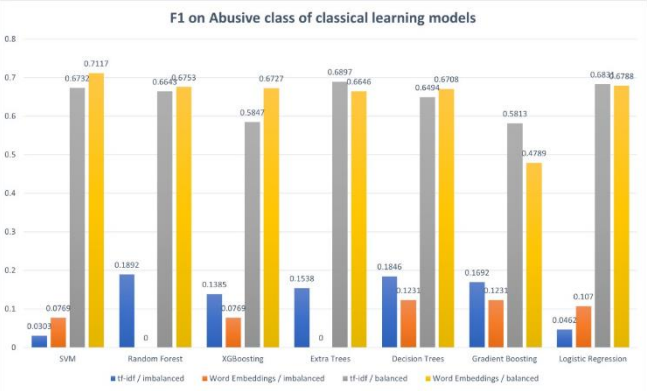
For the unbalanced dataset, tf-idf and word embeddings alternate in terms of best performance. In some models like SVMs, decision trees, and logistic regression, word embeddings outperform tf-idf for the unbalanced dataset. Whereas for models like random forests, extra trees, and gradient boosting, tf-idf outperforms word embedding. However, for the balanced dataset, word embeddings outperform tf-idf for all classical models except for extra trees and gradient boosting. Moreover, the best performing model for the balanced dataset is the SVM model since it has the highest Macro F1-score of 0.73.



Histogram showing the F1-score on the *hate* class only for all the classical learning models and for each of the tf-idf for unbalanced dataset, the tf-idf for balanced dataset, word embeddings for unbalanced dataset, and word embeddings for balanced dataset.

This is most apparent in the F1 score of the abusive class, as it produced very low results for the imbalanced dataset. These results were caused by the low recall scores in the imbalanced dataset. As explained earlier, the thresholds for the abusive class were situated between the normal and hate classes, and this, coupled with the imbalance in the dataset, and the similarities between a hate and an abusive tweet. This was fixed in the balanced dataset. As we can see, the F1 scores of the abusive class in the classical learning model were good using both the TF-IDF, and the word embedding techniques.

For the unbalanced dataset, tf-idf and word embeddings alternate in terms of best performance. In some models like SVMs, random forests, decision trees, and logistic regression, word embeddings outperform tf-idf for the unbalanced dataset. Whereas for models like XGBoosting, and gradient boosting, tf-idf outperforms word embedding. However, for the balanced dataset, word embeddings outperform tf-idf for all classical models except for gradient boosting. Moreover, there are two best performing models for the balanced dataset: the SVM model, and the XGBoosting model both scoring 0.8 in the F1-score.

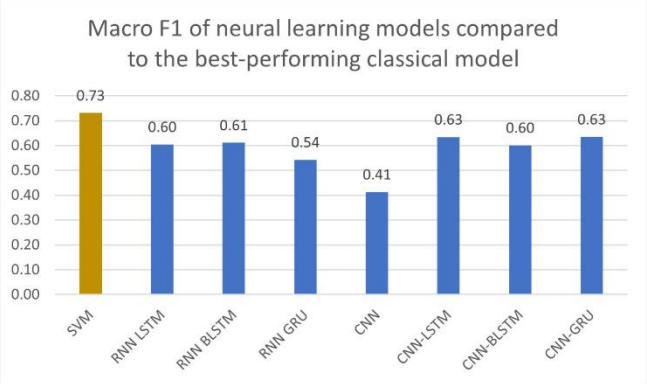


Histogram showing the F1-score on the *abusive* class only for all the classical learning models and for each of the tf-idf for unbalanced dataset, the tf-idf for balanced dataset, word embeddings for unbalanced dataset, and word embeddings for balanced dataset.

After noticing the big inconsistency in results that the imbalanced dataset was causing, we decided to proceed with the balanced dataset only in the neural network models used. For the unbalanced dataset, tf-idf outperforms word embeddings for all classical models except for SVMs, and logistic regression. However, for the balanced datset, tf-idf and word embeddings alternate in terms of best performance. In some models like SVMs, random forests, XGBoosting, and decision trees, word embeddings outperform tf-idf for the balanced dataset. Whereas for models like extra trees, gradient boosting, and logistic regression, tf-idf outperforms word embedding. Moreover, the best performing model for the balanced dataset is the SVM model since it has the highest F1-score for the *abusive* class of 0.7117.

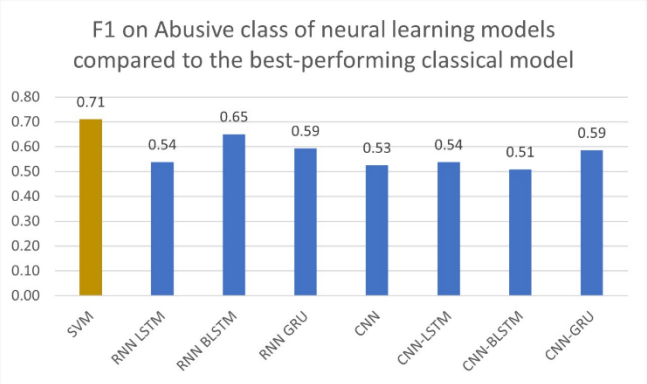
Since SVM scored the highest score for all three different experiments run on the balanced and unbalanced datasets, then SVM can be considered as the best-performing classical model.

E. Results on neural networks models- Balanced dataset



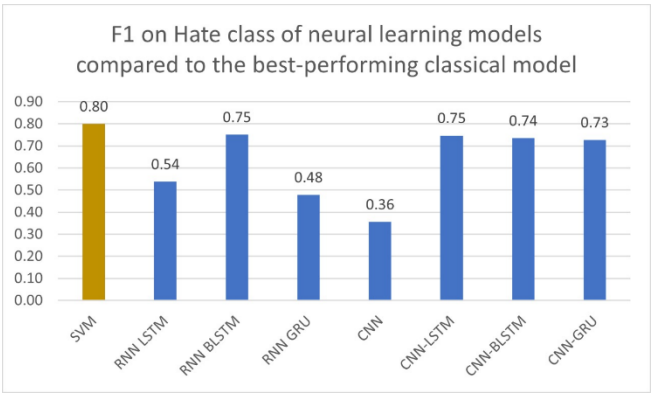
Histogram showing the Macro F1-score for all neural learning models on the balanced dataset compared to the best-performing classical model that is SVM.

The CNN-LSTM model and the CNN-GRU model are the best performing neural learning model since they have the highest Macro F1-score of 0.63 on the balanced dataset. Comparing the Macro F1-score of these best performing neural learning models to the Macro F1-score of the best performing classical model SVM, we get that the highest value is 0.73. This value corresponds to the SVM learning model. Hence, SVM is the best performing model in this case.



Histogram showing the F1-score on the *abusive* class only for all neural learning models on the balanced dataset compared to the best-performing classical model that is SVM.

The RNN-BLSTM model is the best performing neural learning model since it has the highest F1-score of 0.65 on the balanced dataset. Comparing the F1-score of this best performing neural learning model to the F1-score of the best performing classical model SVM, we get that the highest value is 0.71. This value corresponds to the SVM learning model. Hence, SVM is the best performing model in this case.



Histogram showing the F1-score on the *hate* class only for all neural learning models on the balanced dataset compared to the best-performing classical model that is SVM.

The RNN-BLSTM and the CNN-LSTM model are the best performing neural learning model since they have the highest F1-score of 0.75 on the balanced dataset. Comparing the F1-score of these best performing neural learning models to the F1-score of the best performing classical model SVM, we get that the highest value is 0.8. This value corresponds to the SVM learning model. Hence, SVM is the best performing model in this case.

F. Testing cases

Tweet	True label	Predicted (SVM with Embeddings on balanced dataset)	Predicted (Extra Trees with tf-idf on balanced dataset)	Predicted (CNN_GRU with Embedding on balanced dataset)
خراس يلعنلك هالراس واحد حمار دب حيوان	abusive	abusive	abusive	abusive
عندما يأتي اليوم يحترم البنانين بعضهم البعضو عندما تستعمل المنصات لاغراض ايجابيه	normal	normal	normal	normal
كلاب لبنان أحقر من اليهود لاشرف ولامروّة ولادين استقلكم الشعب السوري يوم	hate	hate	hate	normal
بنت ابو ها الله يحميكي ويحمي القائد انتم جبل في جبل ❤️	normal	normal	normal	hate

IV. Comparision to existing work

To understand how well our models performed it is important to compare the results to previous projects by other people who used similar approaches to detect abusive language in Arabic on social media.

In our experimental results it is clear that the classical SVM model is the best performing model with a macro F1 score of 0.73. In our research we found another paper (Kanan et al., 2020) [4] that also used supervised learning techniques to detect cyber-harassment using several models, including SVM. It is important to note that in order to train their models, the paper used a dataset containing around 20,000 tweets and Facebook posts, which narrowed down to around 6,000 after text preprocessing. According to their results, their SVM model was able to achieve an F1 score of 0.947 after preprocessing while ours achieved considerably a lower score of 0.73. However, it is worth mentioning that their papers were based on binary classification. The dataset was consistent of normal, and abusive classes. Thus, they only evaluated the F1 scores of the abusive class which obviously removed the grey area in

the correct classification between abusive, and hateful tweets. Our SVM F1 score regarding the Hate class was 0.8 which is closer to their result. A possible reason to the difference between our scores and theirs could be the fact that we had 3 classes for the model to identify so there is a high possibility that some abusive tweets were labelled as hate and vice versa, which in turn has a negative impact on the F1 score.

V. Conclusion and future Work

As machine learning solutions are becoming more available for variable applications, it is necessary that it's used for the greater good of online platforms, and cyberbullying should certainly be amongst the priorities. Our testing on different models shows that machine learning is indeed capable of attempting to limit cyberbullying on social media platforms, as the ability to classify hate, normal and abusive speech based on training, and achieving reliable and accurate results means that such technologies can cause a great leap into safer social media platforms. We found the SVM model to have the best performance based on training on our dataset, and although the results can be considered accurate, further work and training on a bigger sample size on various social media platforms can lead to greater protection against cyberbullying, an issue that has remained unsolvable since the beginning of the social media era.

VI. References

[1] 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, 117:256–265

[2] 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).

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

[4] Tarek Kanan, Amal Aldaaja, Bilal Hawashin, "Cyber-Bullying and Cyber-Harassment Detection Using Supervised Machine Learning Techniques in Arabic Social Media Contents," Journal of Internet Technology, vol. 21, no. 5, pp. 1409-1421, Sep. 2020.