Arabic Reviews Sentiment Analysis

Youssef Agiza*
youssefagiza@aucegypt.edu
The American University in Cairo

John El Gallab
john.elgallab@aucegypt.edu
The American University in Cairo

1 Experimental Analysis

In this phase we performed a pilot study on different machine learning models to decide the most suitable one for our use case. The chosen models are: Decision Tree, Neural Networks, Support Vector Machine, Naive Bayes, and Logistic Regression. Each of those models is briefly described in the next section along with some of its relevant pros and cons.

1.1 Data Representations

We represented the data representation in three different ways and gave each representation to all of the model to compare their performance. The representations are:

1. **TF-IDF**
2. **Word Embedding**: each word in the sentence is represented by a word embedding vector. All the vectors are then averaged to get one vector representing the whole sentence.
3. **Weighted Word Embedding**: for each word in a sentence, the word embedding vector is multiplied by its frequency in the sentence to get a weighted vector. The vectors of all the words in the sentence are then averaged to give one vector for the whole sentence.

For the word embedding we used the pre-trained embedding provided by the AraVec open source project. Yet, the word embedding provided was not stemmed as the dataset we got from the preprocessing. As a result, we used the raw dataset whenever the word embedding representation was involved in our data. Additionally, we tried normal frequency encoding but TF-IDF gave better results in all the metrics so it was omitted and replaced with TF-IDF. We also attempted doing weighted word embedding using TF-IDF instead of word frequency however, multiplying the TF-IDF vector with the embedding vector didn't work for unknown reasons. One possible reasons may be the difference in Arabic encoding between the two vectors. Another reason may be how sklearn calculates the TF-IDF since we used their library to perform this step.

2 Models Performance

2.1 Decision Tree

The Decision Trees model is suitable for categorical data. This models was not expected to be the best models compared to other models since it is depends on the effect of splitting the data using the features. In our case, the features are the words themselves, which results in high dimensionality of the data. Furthermore, it tries to split the data using the features(i.e. the words) which may result in losing the value of the context in which the words are present.

TF-IDF Encoding. With TF-IDF encoding, Decision Tree yielded an accuracy of 53% which is the highest compared to the other data representation for this model. This result is expected since in TF-IDF the data is cleaned an prepossessed, so we have smaller number of features resulting in more accurate splits.

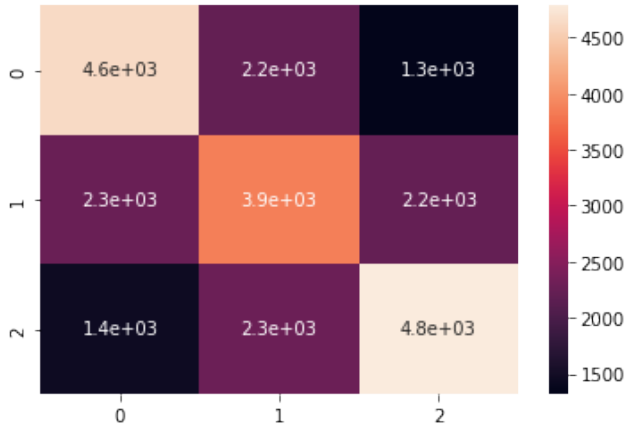

Figure 1. TF-IDF Decision Tree Heat Map

Normal Word Embedding. For the normal word embeddeing, the accuracy of this model is 35% which is very low. However, this is not surprising since we are not expecting this model to be suitable.

Weighted Word Embedding. Finally, the accuracy after applying the weights didn't change, giving 35%.

As expected, Decision Tree results are not promising for this use case. One reason for that may be that the number of features compared to the value gained for each one is not high.

2.2 Neural Networks

Neural Networks(NN) models are based on having several layers of nodes connected together with weights to yield an output in the last layer. NN perform feature engineering automatically as the model learns the optimal weights on its

*Both authors contributed equally to this research.

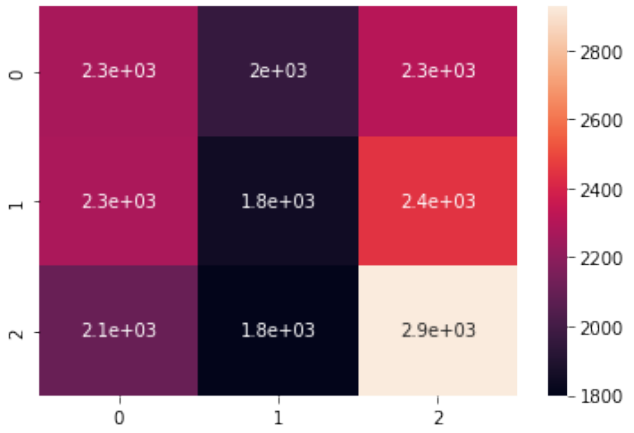

Figure 2. Normal Word Embedding Decision Tree Heat Map

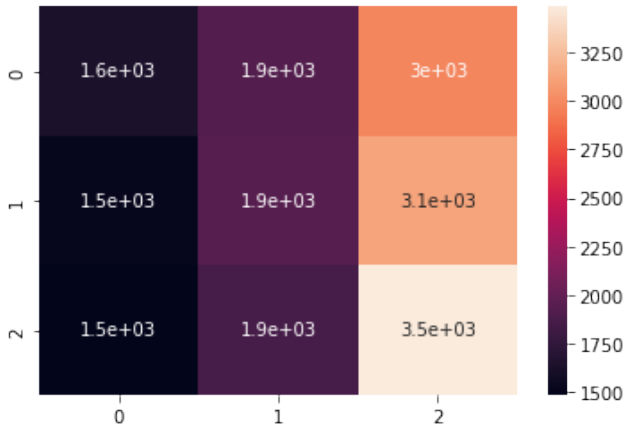

Figure 3. Weighted Word Embedding Decision Tree Heat Map

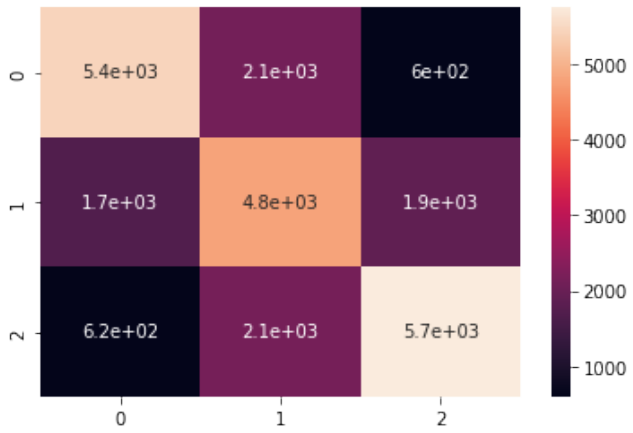

Figure 4. TF-IDF Neural Network Heat Map

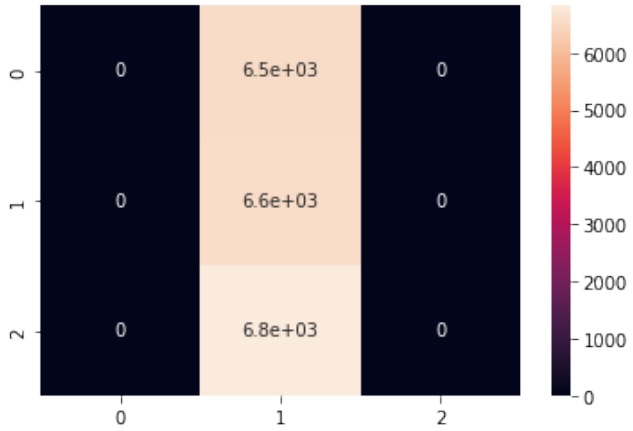

Figure 5. Word Embedding Neural Network Heat Map

own using an error function that needs to be minimized. In this project, NN are expected to be one of the most suitable models since it performs feature engineering to the word and find the right configuration to determine the sentiment of the text. We applied NN using sklearn's Multilayer perceptron classifier (MLPClassifier).

TF-IDF Encoding. As expected, NN yielded an accuracy of 64% which is the highest in this pilot study. The heatmap for the result is shown here.

Normal Word Embedding. The results here were surprising as the accuracy is 33% which is very low especially considering that we are using Neural Networks. Further more, if we inspect the heat map, we can see that the precision, recall, and f1-score are zero for two of the three classes. The only reason we found for these results are that we are treating all the words equally(i.e. unweighted). Accordingly, we tried the third approach: the weighted word embedding.

Weighted Word Embedding. After factoring the weight of the word in the equation by multiplying its frequency with the embedding vector, the accuracy increased to 36% only which is still significantly low. The only reason we could infer for these results was a problem with the embedding itself or the effect of not stemming the dataset.

2.3 Support Vector Machine

Support Vector Machine(SVM) is a linear model that tries to separate the data into classes using a line or a hyperplane. It can solve linear and non-linear problems and work well for many practical problems.

TF-IDF. The accuracy of the model is 60% which is acceptable compared to the other models, yet there are better models.

Normal Word Embedding. Similar to the other models, the accuracy dropped to 34% on applying word embedding. The heat ma

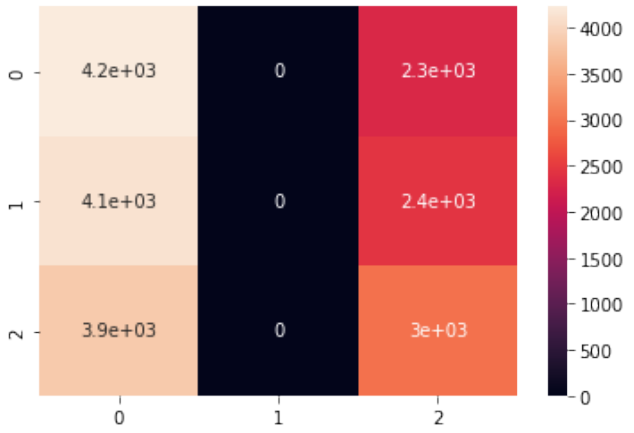

Figure 6. Weighted Word Embedding Neural Network Heat Map

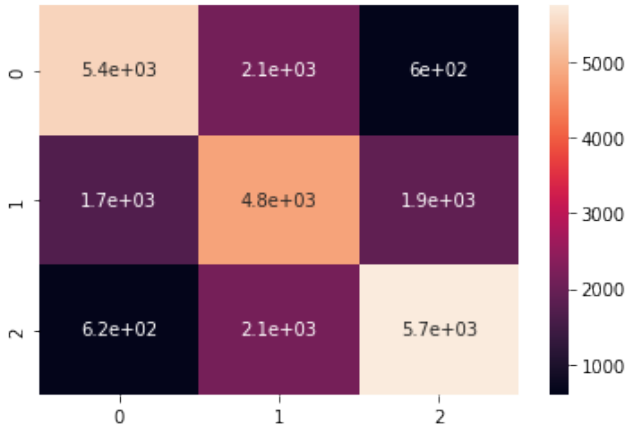

Figure 7. SVM Heat Map

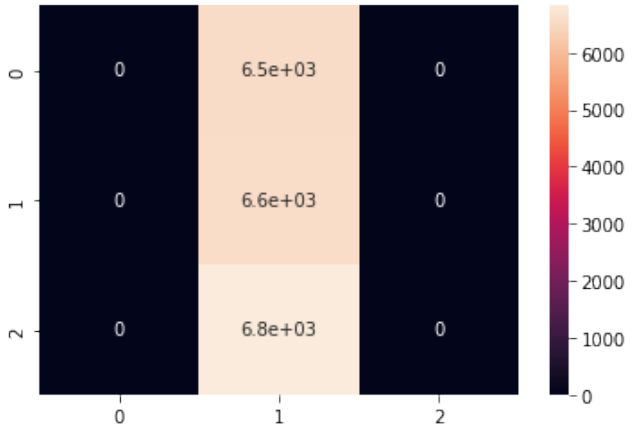

Figure 8. Word Embedding SVM Heat Map

Weighted Word Embedding. The accuracy of the model slightly decreased to 33% percent. Overall, the model accuracy dropped when we used the word embedding.

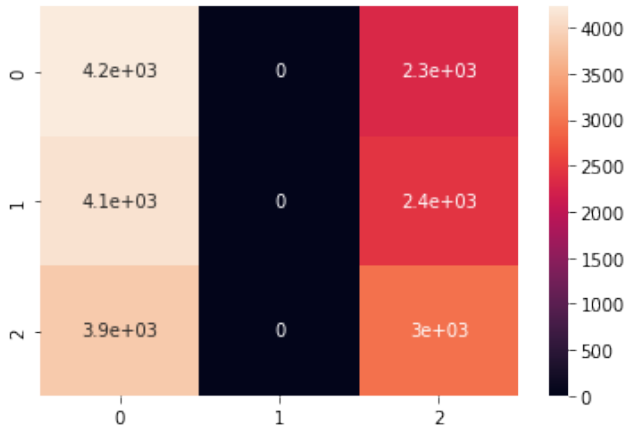

Figure 9. Weighted Word Embedding SVM Heat Map

2.4 Naive Bayes

Naive Bayes is a probabilistic model with an underlying assumption that the features are independent from each other.

TF-IDF. The accuracy of the model is 61% which is among the best we found so far. The heatmap is shown in the figure below.

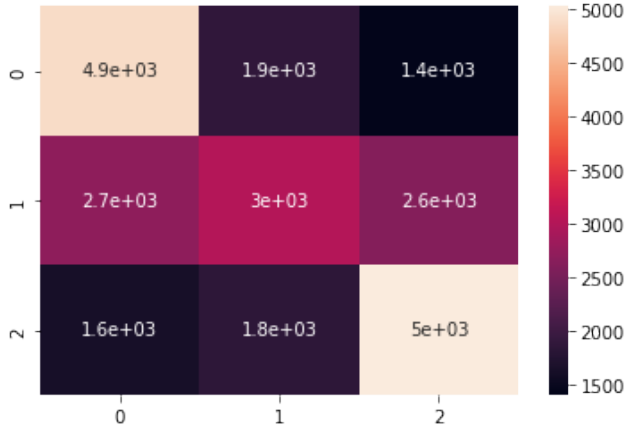

Figure 10. TF-IDF Naive Bayes Heat Map

Word Embedding. The accuracy dropped to 35% similar to the other models. The heat map is given.

Weighted Word Embedding

Again, the accuracy dropped on applying the weighted embedding to 33%. The heat map is given below.

2.5 Logistic Regression

Logistic Regression is a classification algorithm used to predict the probability of a target variable. Yet, Logistic Regression doesn't perform assumption on the data distribution

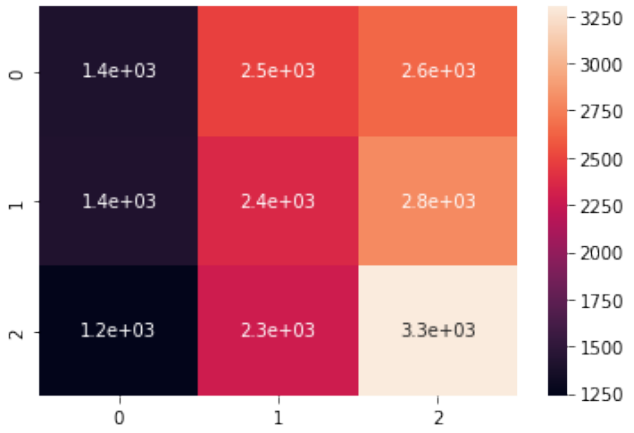

Figure 11. Word Embedding Naive Bayes Heat Map

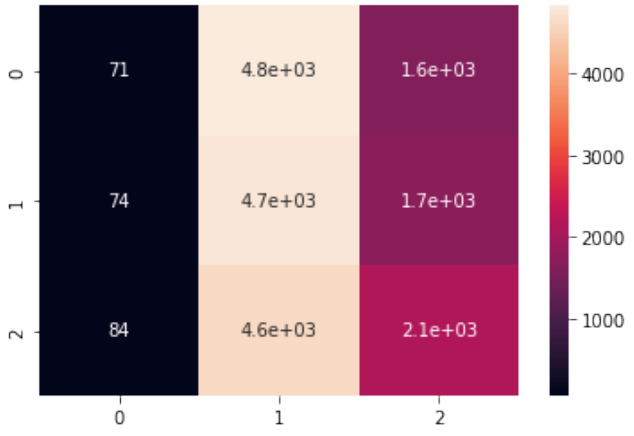

Figure 12. Weighted Word Embedding Naive Bayes Heat Map

unlike Naive Bayes which makes it more generic. Also, One of the main strengths of Logistic Regression is its simplicity. The results of this model were among the best found. One reason for such results may be that the target has a linear correlation with the features which makes since in our case. For example, more positive words are likely to mean positive sentiment.

TF-IDF. The accuracy of the model is 64% which is the best accuracy found so far. This accuracy was achieved by the Logistic Regression Model and Neural Network Model only.

Normal Word Embedding. The accuracy of this model was 35% which was surprising considering the accuracy of this model with TF-IDF.

Weighted Word Embedding. Unlike most of the other models, Logistic Regression accuracy slightly increased to 36% on applying the weights.

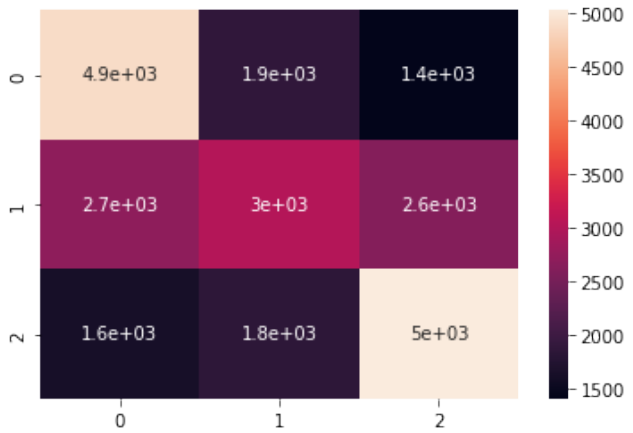

Figure 13. TF-IDF Logistic Regression Heat Map

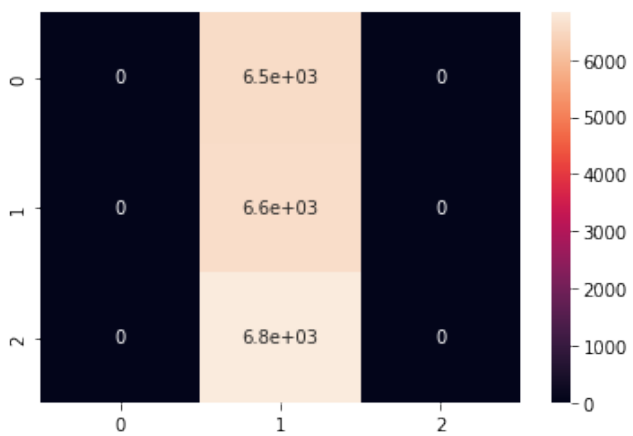

Figure 14. Word Embedding Logistic Regression Heat Map

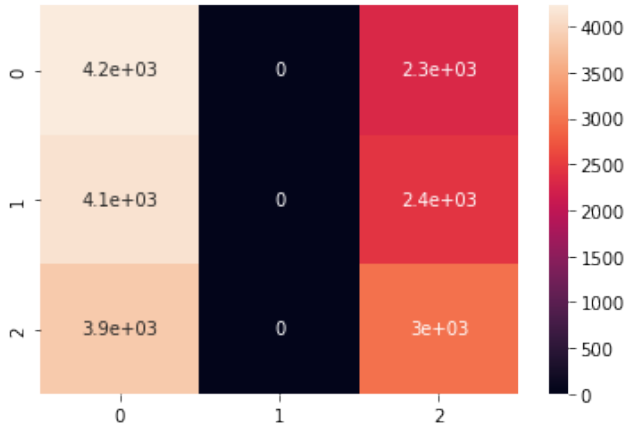

Figure 15. Weighted word embedding Logistic Regression Heat Map

3 Comparison

We can notice a general behavior of the results as the TF-IDF gave the best values in all the metrics. the issue with embedding is not because a normalization for the embedding vectors is needed because we tried normalization and it yielded the same low accuracies. This may either imply that the word embedding approach is not suitable and shall be omitted or that the pre-trained embedding used were not suitable. We assumed that the later reason is more likely to happen since the literature generally shows that word embedding should improve the accuracy.

As already explained, the top three performances are given by the Neural Network, Logistic Regression, and the Naive Bayes.

One advantage that exists in Naive Bayes is being simple and memory efficient. Naive Bayes's assumption of independence between the features makes it computationally fast. However, this assumption may not be valid for our data since different words depend on other words. Similarly, the logistic regression is simple and fast to train. It is also good for generalization since it has no assumptions about the distribution of the class. However, it still assumes a linear correlation between the features and the target which may not always be true. Finally, the Neural Network is the most computationally intensive approach. It requires more memory and time to be trained when compared to the other two models. It may also result in overfitting since the layers may act as memories for the dataset. We avoided this issue by doing early stopping in the training using the 'early_stopping' parameter. Furthermore, we can include word embedding in neural network as part of the training which may solve the issue of the unsuitable embedding mentioned eariler.

4 The Model of Choice

After extensive analysis, the model we decided on choosing is the Neural Network due to several reasons:

1. Neural Networks are flexible as we can configure the layers as we wish to aim for better results.
2. The nature of this model automatically performs features engineering to find the best results. Such approach is useful when trying to predict something as dynamic as sentiment using the text because the natural language usage is not rigorously defined.
3. Word embedding can be integrated with Neural Networks which we expect to improve the results in the following phase.

Using GridSearchCV