Arabic Reviews Sentiment Analysis Model Design

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

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

After the performing a pilot study on several models with different prepossessing operations, we proceeded to design a model for this project. However, the models achieved a maximum accuracy of 64% with some models not predicting full classes as shown in the heat maps. Accordingly, we decided to further investigate the dataset itself in order to find reasons for such result. After some manual investigation, we found that the neutral class(i.e. the class with label '0') had false labels. For example, some text was positive, yet labeled as neutral. As a result we decided to experiment with the dataset after dropping the data labeled as neutral. This decision doesn't affect the problem addresses in our project since our main goal is predicting the positive and negative sentiment of the text, while the neutral sentiment didn't represent an important result.
Surprisingly, the scoring of all the models significantly improved after dropping the neutral labels which encouraged us to completely disregard it and re-perform a pilot study on different models and compare their results. However, for the sake of time and unlike the previous phase, we tested several models only using TF-IDF since it showed highly promising results as explained in the following section.

1 Models Comparison: Updated

In this section we provide a brief comparison between the different models we tried after the change mentioned before.

1.1 Decision Tree & Naive Bayes

Decision Tree scores increased reaching 74% for accuracy, recall, and f1-score of both classes while the precision scored 73% and 75% for the negative class and positive class, respectively. Similarly, Naive Bayes improved reaching approximately 73% for the accuracy, recall, precision, and f1-score for both the positive and negative class. Despite this improvement, these two models have the lowest scores compared to the other models and they are the least suitable for this project. Thus, we mainly consider the next three models.

1.2 Logistic Regression

The values for all the metrics(recall, precision, f1-score, accuracy) of the logistic regression were approximately equal across all the test, so we will be referencing only the accuracy in this section since they all have the same value.
We tested two different solvers, *saga* and *newton-cg*, and they both had equivalent results. Without cross validation, the model scored an accuracy of 84%, while it decreased to 83% on using a 10-fold cross validation.

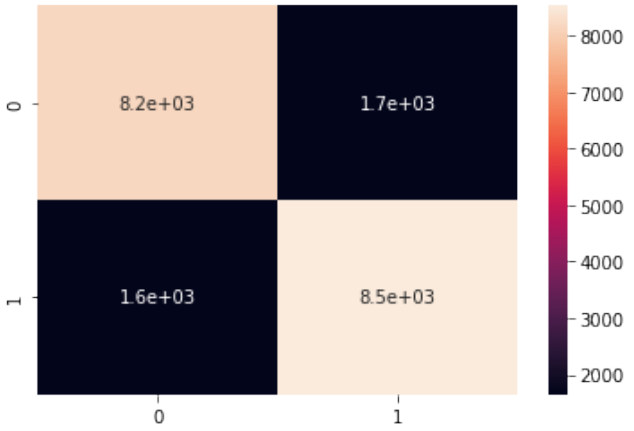

Figure 1. TF-IDF LR Newton solver Heat Map

1.3 Support Vector Machine(SVM)

SVM model improved as well, reaching an accuracy of 81% and almost the same value for f1-score. For the negative class, SVM scored 81% for the precision, and 79% for the recall. For the positive class, it scored 80% and 82% for the precision and recall, respectively. On applying a 10-fold cross validation, all the scores dropped to an average of 80%.

1.4 Neural Network

Finally, the neural network model scores improved reaching an accuracy of 84% without cross validation. The negative class has a recall of 86%, a precision of 82%, and an f1-score of 84%. While the positive class had a precision of 83%, a recall of 87%, and an f1-score of 85%. On applying cross validation, all the scores slightly decreased to approximately 83.7% on average. Thus, this model shows the best values compared to all the other models.

1.5 Chosen Model

According to the metrics and results shown, the neural network has the best values across all the metrics. According to other literature, it is one of the most suitable model regarding the nature of the problem under consideration(i.e. sentiment analysis). It is also suitable for this project due the reasons

*Both authors contributed equally to this research.

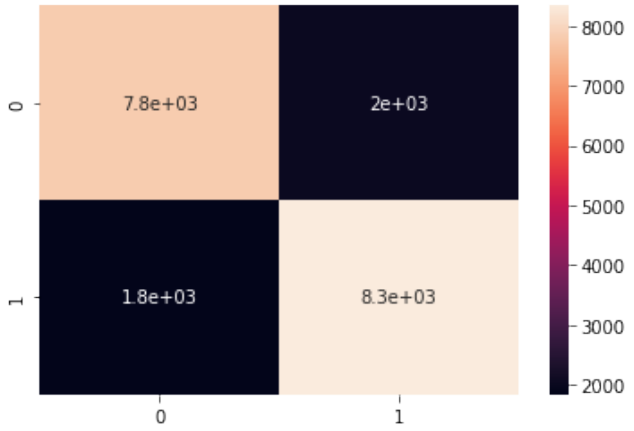

Figure 2. TF-IDF SVM Heat Map

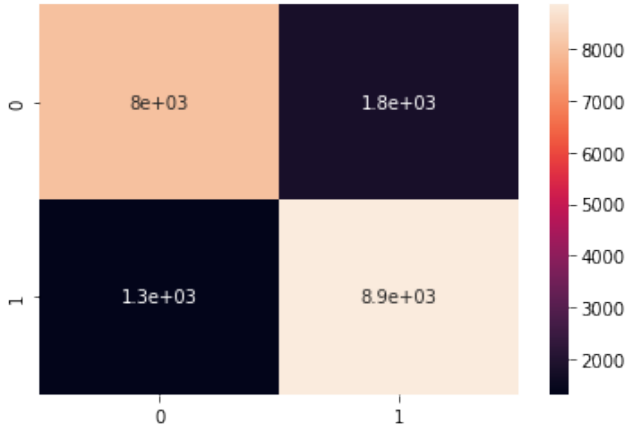

Figure 3. TF-IDF Neural Network Heat Map

mentioned in the previous phase, such as having no assumptions regarding the data and auto-generating features. Due to the mentioned reasons, we decided to proceed with this model for the upcoming phases; and so, we provide a design for the model that we will implement in the next section.

2 Model Design

To decide on the best parameters for our model, we ran grid search CV to compare the results of applying different parameters.

2.1 Hidden layers

Deciding on the number of hidden layers, we tried several values ranging from 2 layers and up to 16 and 32 layers. We found that increasing the number of layers to large numbers didn't result in a significant difference. Yet, a large number of layers is significantly more computationally intensive. Accordingly, we settled on a 3 hidden layers where each of them has 5 nodes. This configuration gave an accuracy of 84% and all the other metrics ranged from 83% to 86%.

2.2 Solvers (Optimizers)

After choosing the number of the hidden layers, we moved on to choose the solver that optimizes the weights. We tested *sgd*, *lbfgs*, and *adam*, each using 10-fold cross validation. *sgd* gave the best results of an average of 83.7% for all the metrics.

2.3 Activation function

We then continued to decided on the activation function. Here, we tried RELU, identity, logistic, and tanh functions. The worst results came from the logistic function which gave approximately 50% for the accuracy and precision, and 30% for the recall. The best results were achieved using the tanh and identity giving an average across all the metrics of 83.9% and 83.7%, respectively. Thus, we decided to proceed with the identity function since it is simpler and the difference is insignificant. For the output layer, we may use sign function to give as -1 and 1 for the output which matches the labels we are using.

2.4 Loss Function

The loss function for sklearn model supports only the Cross-Entropy loss function. Thus, we only experienced with this function at this stage. Thus, we didn't fully eliminate other loss functions. We are planning to research and experiment more with different loss function in the upcoming phase to decide the best loss function.

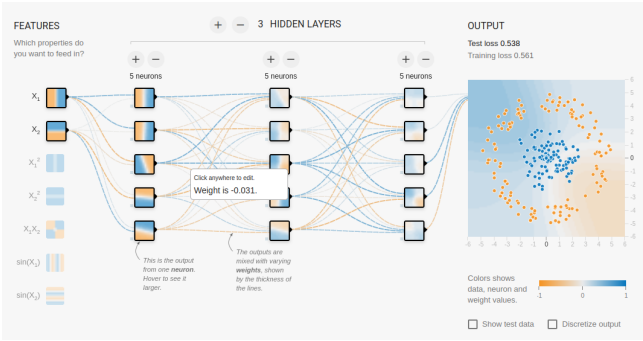

Figure 4. Neural Network diagram

3 Application

The application is supposed to be user-friendly, where the users inputs the features and the application outputs the prediction. In our case, the user shall enter a text through a text field. The text is then sent to the server which will apply the same pre-processing pipeline as that we applied on our data before running the algorithm to give predict the sentiment of the text. The prediction is then sent back to the user through the application. The server will have the

neural network weights stored in the database and all the needed data in order to perform the prediction. We can also improve the model by asking the user for their feedback and use the data to train the model to achieve better results later. The architecture we are going to implement is a client-server architecture. The application is planned to be a web application, but the tools to implement are not fully decided yet. However, we are expecting to use either Django or Node.Js for the backend development.

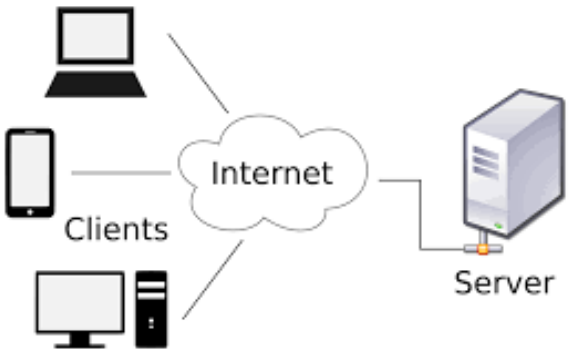

Figure 5. Client Server architecture diagram