**Report**

In this task we worked with the dataset of the Home depot product search relevance competition. Out task was to predict the relevance for each pair listed in the test set. The test set contains both seen and unseen search terms. The provided relevance scores are the average value of the ratings. We wanted to address the distance from the maximum relevance value, so we treated this as a regression task.

We started the task by looking at the input data. We preprocessed the data a bit by merging the train and test datasets with the product\_description table. Also, in order to decrease the dimensionality of the text, we lowered the characters. We also normalized the values, and changed the range of 1-3 to 0-1.

**Character Level Embedding:**

We translated the characters to a unique integer values, using two dictionaries to store the values, in order to turn the search\_term and the product\_description samples into sequences of unique integers. We did see that the product\_description contained a long description, which might be not essential, so we limited the characters to a certain length, and padded the sequences that were shorter.

We created a siamese network, that contain two identical subnetworks and a contrastive loss function joining them. Our model also includes an LSTM and a Dense layers. As for the function, we used the negative manhattan distance function.

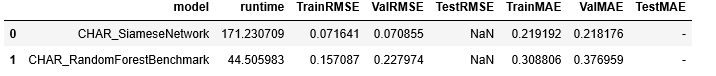
We split the training data into train and validation/test sets. We choose the validation to be 20% of the entire data.

The results we got:



We did submit our results on the test set to the Kaggle competition as a late submission in order to get some indication regarding the results we got. The RMSE we got was 0.56, while the best score in the leaderboard is 0.43192.

We created a benchmark model to compare the results of our model and the benchmark, doing a similar character embedding process as in our model, but here we used the sklearn Vectorizer. The benchmark model that we will use is the Random Forest Regressor.



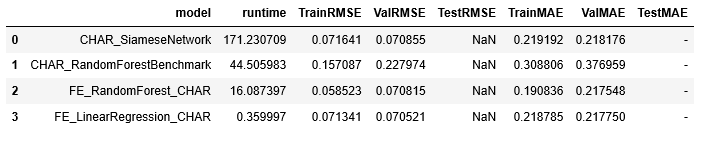
The benchmark model performed better than the siamese model. This showed that our model is not achieving the desired scores to see good enough results.

We thought of a few ways to improve the model:

\* The model is still not tuned with the correct parameters in the intermediate layers. Finding the most precise values for the LSTM node number or the number of outputs in the Dense layer. We tried higher values, but it looked like the model was overfitted when it handles large number of LSTM nodes.

\* There might be some imbalance in the data when we pad the search\_term and the product\_description, the search\_term sequences are a lot shorter than the product\_description, so we need to choose the right amount of characters in each sequence or change the value that we pad with in the padding function, currently its 0.

We checked how the feature extraction abilities of the model compare by taking out the last layers outputs - the processed search\_term and product\_description inputs and concatenate that to feed to a ML model to see the RMSE and MAE of the ML models with the features from our network. The machine learning model we used are the Random Forest model and the Linear Regression model from sklearn.



We saw that the feature extraction ML models had a similar performance as the siamese network. This means that the inaccuracy of our model is in the feature extraction phase, and, maybe, by making the improvements listed above we could have achieved a better score. Unfortunately, due to the lack of time, we did not have a chance to try and implement some of the improvements.

**Word Level Embedding:**

We repeated the process but this time we did the embedding on a word level. As for the rest, we performed a similar process as in the first part.

Here are the results we got:

