UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques i Informàtica

Natural Language Processing

Kaggle Quora Challenge

Brandon Jersaí Alfaro Checa
David Íñiguez Gómez
Jaime Leonardo Sánchez Salazar
Alejandro Vara Mira

Barcelona, April 29, 2024

1 Explanation of the delivery

In this section we are going to briefly explain what the zip folder that we deliver to the professor contains.

First of all, note that we installed the libraries with pip instead of conda because the fasttext library could not be downloaded with conda. Therefore, the *requirements.txt* file will have to be installed using the following command:

pip install -r requirements.txt

Another important thing to highlight, note that our team has been working on the Drive platform. That's why you will see the following lines of code:

Set the HOME environment variable to the desired path os.environ['HOME'] = '/content/drive/MyDrive/NLP Project'
Now you can use os.environ['HOME'] to access the new value home_dir = os.environ['HOME']

This can be found in *reproduce_results.ipynb* and in *train_models.ipynb*. You will surely want to change this path to where you have downloaded your file .zip. Note that the function getwcd() will help you doing that.

Output data: __pycache__, .cache and nltk_data folders. These are folders that were created during the execution of all our functions.

Datasets folder: Here we can see data that we have been generating during the training and testing process.

All other remaining files work as described in the delivery.

2 Outline of the project

The objective of this work is to develop models capable of solving the problem of pairs of Quora questions. This problem arises from the need to identify duplicate questions on the Quora platform. To do that, we have tried different approaches, training different types of models, which we explain below.

As a first approach we chose to apply TF-IDF, since its easy to implement. We obtained good results but not good enough to be considered a good way to solve the Quora Questions Pairs problem, since we are using a very simple algorithm.

After that, we considered building embeddings for each sentence. That is why we decided to use Fast Text library. First, we decided to use a supervised algorithm that includes text classification. We got better results than TF-IDF. we also tried to optimize its hyperparamters. Unfortunately, we could not obtain better performance It may be due to the fact that the model is unbalanced having more examples corresponding with one label than the other. If we had a balanced dataset, the model would be unbiased, so the hyperparameter tuning would have been ore effective.

Then, we decided to add extra features, hoping we get better results. To do so, we used different metrics and ratios and the unsupervised algorithm of Fast Text, building embeddings for the sentences. After that, we computed distance between the embeddings, and adding those extra features we trained two classifiers, Random Forest and XGBoost. Since we are using an unsupervised algorithm and a classifier that is not as synchronized as the supervised method of Fast Text, we could not either improve any evaluation metric shown above.

Metric	**Accuracy**	**Precision**	**Recall**	**F1-score**
TFIDF	0.738	0.719	0.713	0.716
Fast Text Supervised	0.792	0.7904	0.792	0.791
Fast Text Supervised Hyperopt	0.773	0.77	0.773	0.765
XGBoost Feature Extraction	0.715	0.719	0.715	0.717
RF Feature Extraction	0.698	0.718	0.698	0.702
MiniLM-Baseline	0.780	0.797	0.780	0.784
MiniLM-05	0.892	0.895	0.892	0.893
MiniLM-03	0.89	0.895	0.89	0.891
MiniLM-05-Online	0.894	0.897	0.894	0.895
MiniLM-03-Online	0.892	0.895	0.892	0.893
Distilbert-Base	0.907	0.907	0.907	0.907

Table 1: Performance Metrics for Different Models

Finally, we used a more sofisticated group of models, that are Sentence Transformers. These are state of the art neural networks that can perfectly build embeddings for sentences. We tried two models, one more general and a pre-

trained on a custom dataset.

As we can see, finetuning a model makes it perform better on custom datasets, since for our problem we don not need a model to be as general as possible, but to be able to do correctly a very specific task. One can notice the 10% accuracy increase in all variations we made to de baseline model. With respect to modifying the margin, we could see that there are no big improvements made. If we did not modify the threshold with the validation set, the model with the modified margin would perform better. It is reasonable to think this because the threshold of the model with the modified margin is higher, which means that this model would have better classified the examples with a probability around 0.5 because it would have 'dragged' them towards a distance greater than 0.5 and that if we had considered the predefined margin this would not have happened.

Relative to the choose of the loss, we can see that the Online Contrastive Loss makes the model perform slightly better than with the Contrastive Loss, as one could expect because this loss focuses on hard examples.

Finally, using the Distilbert quora pretrained model without fine-tuning it showed an impressive performance, geting an accuracy greater than 90%. Compared to the MiniLM Baseline model, we can see that the fact that the model is pretrained with a custom Quora Dataset makes the model perform much better than a general model. It is also relevant that the Distilbert model maps the questions to a higher dimensional space, and also that it is probably trained on a bigger dataset, so it captures better complex relations between sentences or words.

3 Contributions of Each Team Member

Brandon: Responsible for developing a **TF-IDF vectorizer from scratch**, encapsulated within a class named TFIDF. The construction process involved several key steps, which are described in the *utils_BrandonAlfaro.ipynb* file. These vectors were utilized to train a Logistic Regression model, which was fine-tuned with optimal hyperparameters through GridSearch. The model was then fitted using the training set and evaluated on the validation set to assess its performance, while also finding the optimal operating point on the ROC curve, the one which would maximize the difference between TPR-FPR.

Upon obtaining the optimal model, it was evaluated with the chosen operating point (threshold) using the test set to ensure its generalization capability. Also, a classification report was displayed, which included the main metrics in classification: accuracy, precision, recall, f1-score.

Following thorough validation, the model was further refined using the entire dataset, including the training, validation, and testing sets, to train a final iteration optimized for predictive accuracy. This final model was deployed to make predictions on the Kaggle dataset, leveraging the training it received from the merged dataset. By incorporating information from all stages of model development, the final model aimed to achieve the highest possible predictive performance on unseen data from the Kaggle competition.

David: Responsible for finding a solution to the Quora Question Pairs problem using **Sentence-Transformers**, that is a Python framework for state-of-the-art sentence, text and image embeddings. We build sentence embeddings and then measure the cosine distance between sentences, so that if sentences are duplicated, the distance is close to 1, and to 0 in the other case. For more information, please have a look at *utils_DavidIniguez.ipynb*.

I looked at several web pages looking for more information about it, and which evaluation and loss functions to use. I finally decided to use the **BinaryClassificationEvaluator** evaluator and the **ContrastiveLoss** and **OnlineContrastiveLoss losses**. Related to the models used, we show 2 pretrained models (one general model and a more specific one trained with a similar dataset) and several variations of the first one, depending on the values of parameters of the loss, and the loss function that is going to monitor the convergence of the fitting algorithm. Due to memory issues, I could not perform any fine tuning on the second model, since it's a much bigger model.

With all these models, I built a dataframe in order to compare the performance of the different models, based on the predictions of the models on the test dataset, unseen for them until the predictions.

To optimize the predictions, I built a function that finds the optimal distance threshold. To do that, we build the roc curve, and find the operating point, that is the threshold that maximizes the accuracy of the validation dataset. Finally, with this threshold, I developed another function that predicts the test dataset and plots the confusion matrix and a classification report. It also gives the opportunity to plot the confusion matrix for the validation set with the new threshold and to find the optimal threshold for the test set, but I found them a bit irrelevant, so I did not show them on the *reproduce_results.ipynb notebook*.

Jaime: Responsible for implementing a solution to the problem using embeddings. More specifically, Fast Text has ended up being used, training it based on the questions (from the train) that we had available.

Prior to this approach, an attempt was made to calculate embeddings from pre-trained models of Word2Vec ("word2vec-google-news-300") and Fast Text ("cc.en.300.bin"). Once the embeddings for the questions were calculated, the idea was to implement certain distances between embeddings in order to train a

model (such as Random Forest or XGBoost). Despite not having worked with all the available data, tests were carried out (with a reduced data set) achieving an accuracy of around 68%.

With this type of models, it took too long to compute the embeddings for all the questions, so it was decided to change the approach. This time, as mentioned above, training Fast Text with the Quora Challenge questions and training a supervised classification model using these embeddings to make predictions about new text instances.With this approach we improve the prior accuracy we got implementing TF-IDF.

With the intention of continuing to improve the obtained results, we chose to use Fast Text again but this time training a model in an unsupervised way. That is, now simply obtaining the embeddings, from which distances functions and Feature Extraction functions will be applied with the intention of training a model (such as Random Forest or XGBoost). Unfortunately with this approach we did not improve the previous accuracy.

Alejandro: Responsible for preprocessing all the questions by cleaning them. I expanded contractions and abbreviations, removed punctuation, performed spellchecking, removed stopwords, and removed accents using different functions. Regarding spellchecking, I tried two approaches: one using the pyspellchecker library and the other using a BKtree. Both were too computationally expensive, but they can be found in my personal utils.

Then, my focus was redirected to extracting new features from the cleaned questions. For each pair, I computed:

1. First Word Equal: Checks if the first word in both questions is the same.
2. Common Words Ratio: Calculates the ratio of common words between the two questions.
3. Flesch Reading Ease (Question 1): Measures the readability of Question 1 using the Flesch Reading Ease formula.
4. Flesch Reading Ease (Question 2): Measures the readability of Question 2 using the Flesch Reading Ease formula.
5. Flesch Kincaid Grade (Question 1): Assesses the grade level required to understand Question 1 using the Flesch-Kincaid Grade Level formula.
6. Flesch Kincaid Grade (Question 2): Assesses the grade level required to understand Question 2 using the Flesch-Kincaid Grade Level formula.

After this, with Jaime's help, we trained an unsupervised fasttext model with the questions, then computed all their embeddings and calculated distances (cosine, Euclidean, Manhattan) between pairs of embeddings and considered the

distances as new features.

Finally, with all these features, I trained a random forest classifier and an XGBoost classifier, with an accuracy close to 70% in both.