Natural Language Processing Project

Quora Questions Pairs

Authors:
Cécile Ledoux
Anabelle Lichou
Maxime Minard

2th May 2022

1 Introduction

The goal of our project is produce an algorithm that predicts whether a question is a duplicate of another. Questions are given to the algorithm as pairs and the algorithm must output a value in the interval $[0, 1]$ representing the probability that the two questions are duplicates. Evaluation of the model is done with the the log loss between the predicted values and the ground truth.

2 The Dataset

Our dataset is composed of 404290 pairs of questions. Questions have a mean length of 119 and a mean number of words of 57. 3 lines have null values and we decided to delete them.

The classes are slightly unbalanced as the proportion of pairs with the same meaning is only 36%.

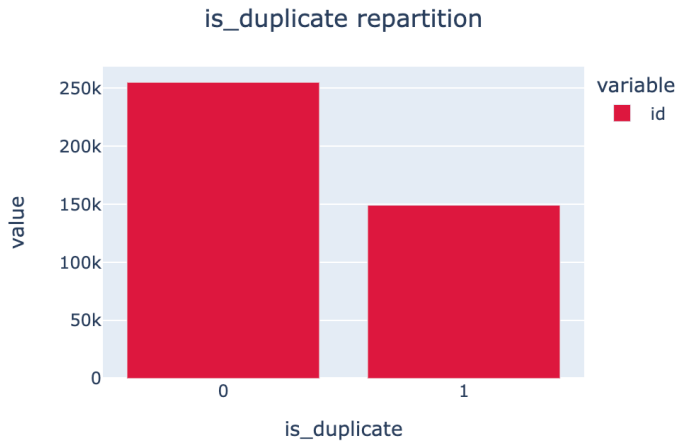

Figure 1: Labels repartition within the data set

As it can be seen in this WordCloud, the most common words are those usually used to ask questions ("best way","difference") but also some specific subjects ("make money", "Donald Trump"...).

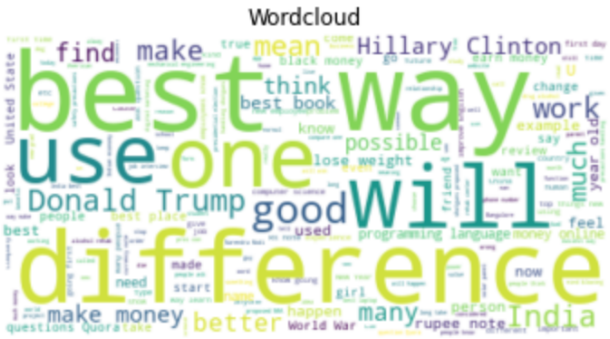

Figure 2: Wordcloud

3 Features selection and data visualisation

To have more insight on our dataset, we tried to extract significant features for the classification.

3.1 Basic statistics

First, we looked at the structure of the sentences. We ploted the length of the questions and the number of words for each question of the pair. It appears that similar questions are a little bit shorter than others.

We also computed the difference of length and of number of words of the 2 questions of the pairs. It is not surprising to see that duplicated questions have a closer structure than others. Indeed the mean difference of length and of number of words is quite close to zero.

Is duplicate	Questions Length	Diff. Questions Length	Nb words	Diff. Nb words
No	128	-0.84	15	-0.19
Yes	105	-0.09	12	0.001

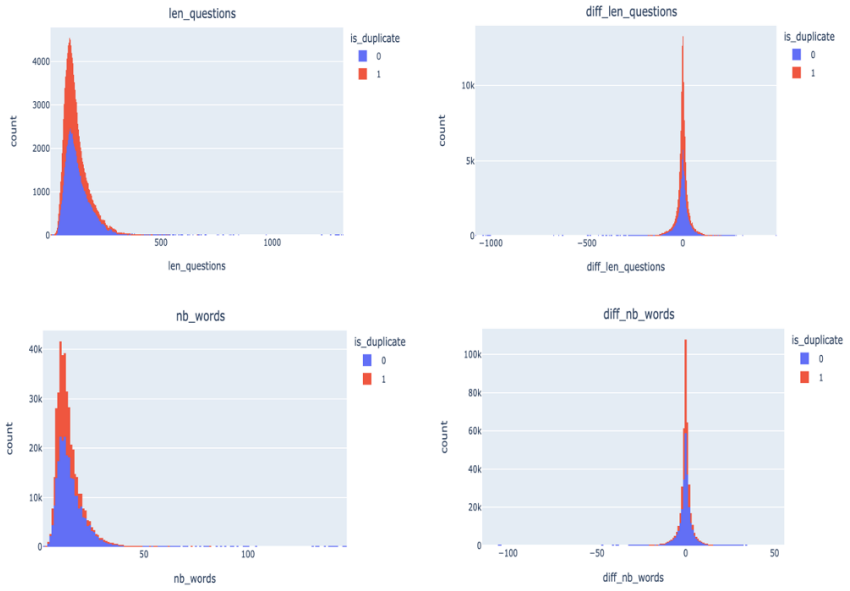

Figure 3: Basic features

3.2 Word share

Another feature that gives insight on the similarity of two sentences is Word Share which is the number of common words divided by the total number of words.

To compute this feature, we removed the most common words to select only those which really show the meaning of the question.

As expected, similar questions have an higher Word Share ratio.

Is duplicate	Nb. Common Words	Word share ratio
No	2.21	0.16
Yes	3.42	0.28

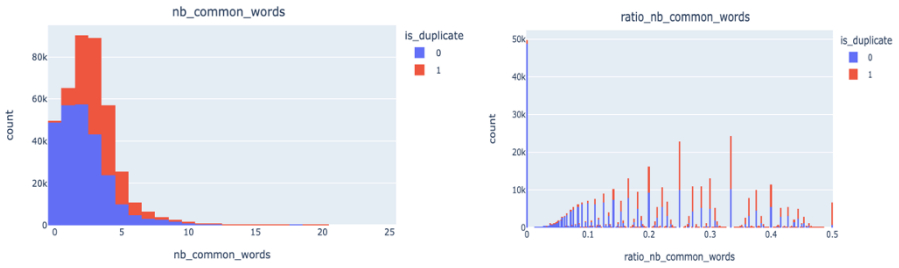

Figure 4: Basic features

3.3 Advanced features

Now, we can use more sophisticated computations to extract features.

The first one is the Levenshtein distance which is equal to the minimum of characters that should be added or deleted to go from one sentence to the other.

We also used fuzz ratios which are inspired by this distance. Simple fuzz score is 100 minus Levenshtein distance. The Fuzz-partial-ratio is the score of the shortest word compared to the most similar word of same size. Fuzz-token-sort ratio compares sequences in disorder and Fuzz-token-set-ratio divides the sequence in three, compute the ratio on the three and keep the greatest.

Finally, we computed Sequence Matcher score which is the longest contiguous matching subsequence.
Here are the scores we get :

Duplicate	Lev. distance	Partial-ratio	Token-sort ratio	Token-set-ratio	Seq. Matcher
No	46	60	59	67	0.51
Yes	26	72	72	82	0.68

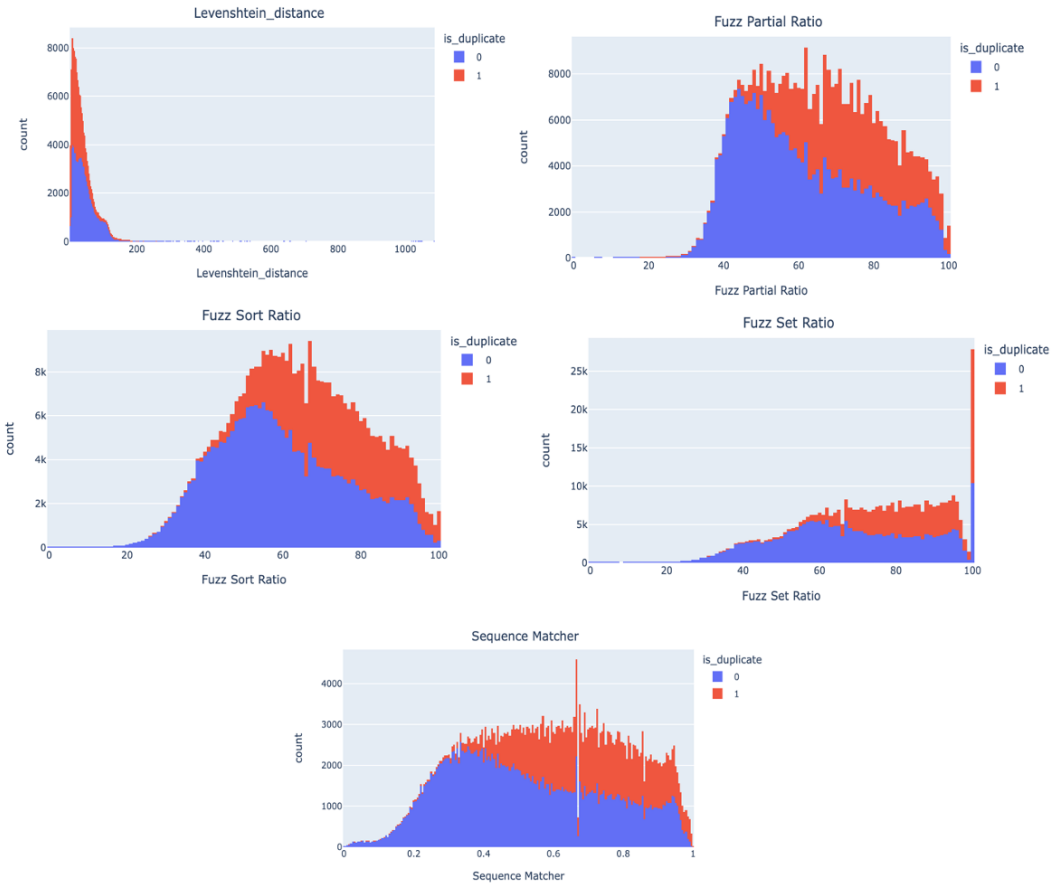

Figure 5: Advanced features

4 Data pre-processing

4.1 Cleaning data

The first step to process data is to clean it. We began by removing all characters which are not letters, numbers or punctuation, by specifying common abbreviations and by lowering words. We also removed "stop words" which are very common words which do not have an influence on the meaning of the sentence (e.g : "and","by"...).

Then, we tokenized each question by separating it into words.

Finally, we lemmatized each word. Lemmatization applies a morphological analysis to a word and returns its dictionary form (e.g : "meeting" becomes "meet").

4.2 Transforming questions into vectors

The next step is to transform the sequences into vectors. We used a tokenizer to set an index to each word and each sequence is represented by the index of its words. In order for the sentences to have the same length, we padded the sentences.

4.3 Word embeddings

To finish the data processing, we transformed words into vectors using word embeddings. Word embeddings maps words into space in such a way that similar words are close and that the geometry of space is meaningful.

To be more precise, we choose to use Glove. This method is based on a co-occurrence matrix that is to say the number of times a word appears in the same sentence as another.

5 The model

5.1 Siamese LSTM

The first model we tried is siamese LSTM. It combines two concepts: Siamese networks and LSTM.

Siamese networks are networks that have two or more identical sub-networks. Given a pair of similar examples (x_i, x_j), a Siamese neural network is a network that uses the same weights to compute two outputs (h_i, h_j) (one for each example). A Siamese network is trained to output similar vectors h_i and h_j for similar input examples, and dissimilar outputs otherwise. They work well on similarity task and have been used for instance to recognize forged signatures or in face recognition.

LSTMs were developed to deal with problems encountered with traditional RNN such as vanishing gradient and sensitivity to gap length.
Our model takes the two questions of each pair as inputs. Then each question goes through a non-trainable embedding layer to compute word-embeddings and through a 64 LSTM layer. Finally, the two inputs are concatenated and two hidden fully-connected layers are applied before making the final prediction.

As our model is overfitting very quickly, we kept the number of layers and of parameters low and we added dropout.

5.2 Inprovement of the first model

To improve our model, we had the idea to present the words which are common to both questions and those which are not as the two inputs of the network. By doing so, we give additional information to the network about the similarity of the sentences. We also wished to make the network weight differently words when they are in common or not as it could be have more a very difference inportance in those two contexts.

5.3 Analysis of features

We also wanted to create another model which takes into account the structure of the questions.

To create it, we used the features that we had extracted (Word Share, Fuzz ratios...) as inputs of a fully-connected neural network with a 32 neurons hidden layer.

5.4 Combining the models

The final model combines all the networks previously described (without the last layer). It has 5 inputs (question 1,question 2, common words, differents words and extracted features). After

those features passed through their network, they are concatenated. The resulting matrix goes through two fully-connected hidden layers before making the prediction.

6 The results

As it can be seen in the graphs below, the LSTM model based on common/different words and the final model have by far the best results. As the final model overfitts less in comparison and consequently offers better performance on validation set, we decided to keep it. We stopped it at epoch 10 when it begins to overfitt. We obtained a loss of 0.3194, an accuracy of 83 and a f1 score of 80 on validation set.

The model which uses features as inputs only learn during one epoch.Its score is quite low comparing to the other networks but it is still an improvement compared to a naive approach.

LSTM model with the questions as inputs has better results than the feature networks but significantly worst performance than the final model.
Here are the results on validation set :

Model	Loss	Accuracy	F1
Final model	0.32	85	81
Feature model	0.50	71	67
Questions LSTM	0.42	80	73
Common/Diff LSTM	0.36	83	78

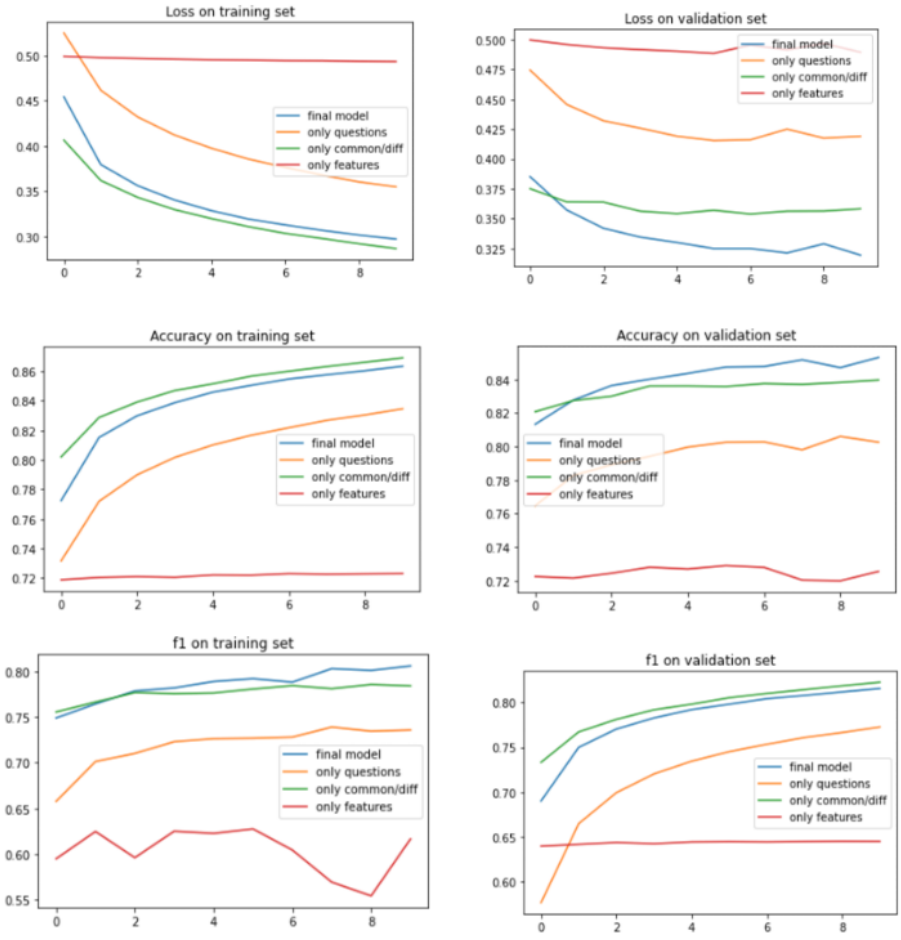

Figure 6: Results