Altegrad Challenge : Report

Elliot Hofman (Aylliaute H)

February 6, 2018

In this Altegrad challenge, we were asked to implement a model to detect whether two short questions have the same meaning or not. The data has been collected on the Quora website (around 80 000 samples in the train set and 20 000 in the test set). Each element of both the train and the test sets consists in a pair of questions, and the target to predict is a binary value to indicate whether the two questions are duplicate or not.

Contents

I - Preprocessing

The first step when dealing with textual data is to implement a preprocessing function to normalize the data as much as possible, so as to reduce the noise. I wrote a function strip_accents_unicode to deal with accents and the few characters with weird encoding. Then I normalize all forms of english contractions I could think of, I replace all numbers such as '3000', '25.000', '6,000,000' etc to '3k', '25k', '6m', I replace symbols such as '' or '?' to 'euro' and 'rupee' and I correct typos and regroup common expressions that came to my eyes.

At that point, I remove from the questions everything that is not alphanumerical (including punctuation), set everything to lowercase and willingly keep stopwords (I believe they still carry on predictive value given the task we are supposed to solve).

Then, I use NLTK to tokenize questions. I personally do not like stemming too much, and I further search for the word embeddings in the word2vec, gLoVe and fastText corporas, so I need my normalized words to remain existing words. That's why I use instead the WordNetLemmatizer to set verbs to their infinitive form, and the inflect package to set plurials to singulars.

I finally came up with a quite satisfying normalization of questions. Some examples :

```
India: Which are the most common Marathi phrases/ expressions used in everyday life?
india which be the most common marathi phrase expression use in everyday life

My boyfriend has been cheating on me with girls that are in a game called dark legends and
been talking to them on kik what do I do?
my boyfriend cheat on me with girl that be in a game call dark legend and talk to them on
kik what do i do

Who are the best personal financial advisors in Green Bay for people saving for retirement?
What were your experiences with them?
who be the best personal financial advisor in green bay for people save for retirement what
be your experience with them
```

II - Feature engineering

(1). NLP based features

Those are the features I obtained directly upon the questions content.

question_flags : For each pair of questions, I determine which is the question flag. The intuition behind this is that some pairs of flags might strongly indicate non-duplicate ('how long'/'who', ...), whereas maybe some particular pairs are statistically very likely to be duplicates among the dataset. So I collect all of the possible flags pairs, stock them with one hot encoded representation (leading to some sparse encoding) and perform a chi-2 test to select the 25 most predictive pairs of flags.

last_character : Weither questions in the pair ends with the same character

embedding_metrics : A bunch of statistics made on cosine similarities between all pairs of word embeddings of the two questions (mean, median, standard deviation, kurtosis). This is done only with non-stopwords tokens and with all three of w2Vec, gLoVe and fastText embeddings.

shared_words : Common n-grams between pair of questions, with n up to 8. This is done with both processed and non-processed versions of the questions.

question_lengths : Characters and words counts, along with absolute difference between pairs of questions?

levenshtein_distance : Levenshtein distances between pairs of questions, done with processing and without processing

tfidf_stacked : Some instances of scikit TfidfVectorizer and CountVectorizer with range of (1, 3) grams and no limitation on the vocabulary size are fitted on the whole corpora and each pair of questions is transformed into the TFIDF/BOW space so as to form the absolute difference between the tfidf/bow vectors of q1 and q2. Then Multinomial NaiveBayes, Logistic Regression and Random Forest are trained and stacked (out of k-folds strategy, k = 10) to embed the very sparse TFIDF/BOW space into a 1 dimensional feature.

gpe_coherence : Spacy is used to get NER tags about locations. Then three columns are created to indicate whether the questions are GPE-concordant, GPE-discordant or if the feature is irrelevant (no GPE-tags were found in the two questions)

words_mover_similarity : Words mover distance is calculated with the word2vec embeddings. Distances are then post-processed so as to form a similarity measure.

fuzzy_features : A bunch of string similarities calculated with the fuzzywuzzy package.

spgk_feature : Implementation of the shortest path graph kernel between pairs of graph of words representations of the questions.

A few comments :

All the features are encoded so that they do not depend on the order of the questions in the pair (for instance, (q1_length, q2_length) is transformed into (Min(q1_length, q2_length), Max(q1_length, q2_length)).

Some features (words_mover_similarity, spgk_feature, ...) were readjusted by hands. In fact, as the similarity increases the ratio of duplicates increases aswell, until similarity gets very high and then the duplicates-ratio decreases. This is because some questions are very similar (seen as strings) but a single word difference changes the whole meaning. So I coded handcrafted rules (based upon whether questions differ from a single word or not) to readjust those features so as to force a linear dependence on the target. The explanation behind this is that I used first logistic models, and I then switched to Gradient Boosting methods able to learn non-linear patterns so this post-processing is not useful anymore but I kept it anyway.

According to NER features, I thought the idea was great. I wanted to identify for each pairs of questions every NER tags and use them to create boolean features that indicate whether questions are concordant or discordant according to the GPE-tags, PERSON-tags, ORG-tags. The problem was that the Spacy package is not so accurate at tagging NER's and often does not detect or outputs very wrong tags. I wanted to retrain the Spacy model on the corpus, but it required an unreasonable time of hand-labeling. An other solution was to use the NER-tagger to parse the whole corpora in a first round, and then, when creating the features, to store for each token the most frequent NER-tags associated. I tried it but the NER-tagger was still not accurate enough to me. An other problem of NER tags is the noise in the labels : some questions as 'Is it safe to raise a child in Nigeria ?'/'Is it safe to raise a child in Belize ?' are marked as non-duplicate, whereas others such as 'What safety precautions about guns in Arizona ?'/'What safety precautions about guns in California ?' are marked as duplicate. That's why I found NER-tags dangerous to use in the model. Finally I corrected and implemented handcrafted rules about GPE-tags, but was not brave

enough to make it for other kinds of tag (this is long and basically not machine learning anymore)

Window-size and depth for the SPG Kernel are set to 2 and 1. A greater window-size would break the information in the graph of words I thought, because all of the questions are very short. This led to a nicely target-correlated feature.

TW-IDF with closeness weights were implemented and tried to replace the TD-IDF vectors. It did not gain on the LB, and was also somehow long to fit on the corpora (the implementation is not as optimized as scikit vectorizers). So I decided to remove them from the model.

(2). Structural features

Those are features obtained upon the construction of the graph formed by the pairs of questions.

magic_features : For each pair, determine the frequency of the questions among the whole corpora, and also the numbers of common neighbors they share (with how many common other questions they appear in the corpora). The official Kaggle Quora challenge refers to it as magic because they are highly predictive and can seem leaky but I believe they are not (if a question is very frequent it is reasonable to think it is more likely to be duplicate, etc)

graphical_features : Once the whole graph of questions is built, compute some features based on K-cores, centrality, closeness and number of cliques (calculated with the package networkx)

III - Model

(1). Light GBM

The NLP and structural features are merged together and fed to lightGBM. LightGBM is a nice alternative to XGBoost when one wants to speed up training time because trees are grown leaf-wise instead of level-wise. First I trained scikit models such as Logistic Regression, but I finally got best results with lightGBM. I think this is because lightGBM really learns to push probabilities so as to minimize the logloss (setting very confident predictions to 0.9999 for instance), because differences in accuracies were not so significant. I optimized only two parameters, namely the learning rate and the max_depth of trees. I did this with the package BayesianOptimization, which computes at each iteration a bayesian posterior estimation of the function to optimize (here a wrapper around the cross validation results of lightGBM) and hence automatically finds the best ranges of parameters to investigate. Finally, I set the number of trees to a very high value and use some early stopping validation to prevent overfitting.

(2). Keras LSTM

Here I borrowed the scripts of the solution https://github.com/aerdem4/kaggle-quora-dup. The idea was to mix my shallow lightGBM model with an other deep approach. This nice solution consists in a fairly simple structure : the two questions are encoded separately by the same embedding/LSTM layer (Siamese model) and merged along with some other features in various dense MultiLayerPerceptrons. It also has the nice idea to extract the number of rare words in common between pairs of questions (words that appear less than 100 times in the whole corpora, which I think is very predictive). I made it a small change

however : The output of Ahmet's Siamese network encodes $(x+y)|(x-y)^2$ where x and y are the sequence obtained by the shared LSTM layer for the first and second questions. I understand that he wanted to keep the model absolutely symmetric, but I thought anyway that I would rather break the symmetry and consider directly the concatenation of x and y because I suppose that the sum makes a lot of information disappear. So I changed $(x+y)(x-y)^2$ to $x|y|(x-y)^2$. I also modified the code so that it used my own pre-processing function, that I thought better than Ahmet's one. Simply modifying those two things gave me a public LB score of 0.13375 and a friend of mine told me he had found and submitted the same code without any modifications and got a score of 0.137.., so the modifications were indeed fruitful. Finally, I trained the modified code with gLoVe, but also fastText and word2Vec embeddings.

(3). Stacking

The two models are trained on 10 Stratified Folds and the predictions are averaged to make the models more robust (more generalization). Finally I mix the predictions of the LSTM and the predictions of lightGBM giving a weight of 0.6 to the LSTM and 0.4 to lightGBM. This gained a lot on the leaderboard, LSTM alone had 0.13375, lightGBM alone had 0.136.. and with this strategy I got to 0.12734. I tried to stack with other models, such as LogisticRegression, extraTrees, and the LSTM trained with fastText and word2Vec embeddings, but it made no impact on the final score.

(4). Post processing

Finally, I used Ahmet's post processing function to boost my scores. I changed the thresholds to suit better my probabilities, and also removed the calibration of class weights part (the official kaggle challenge had unbalanced distribution between the training and testing set, which was not the case for the Altegrad challenge, I submitted a solution full one 1's and deduced the test distribution, just to realize it was absolutely stratified with our training set). In a nutshell, it uses the knowledge of true duplicates and non-duplicates we have from the training data to boost the model probabilities (it gains some confidence about a pair being duplicated and set the probability 0.7 to 0.99, and conversely it gains some confidence about a pair being non-duplicated and set the probability 0.3 to 0.01, etc). This also had a great impact on public LB.

IV - What I wanted to try and did not implement

There are a few things that I wanted to try but did not implement.

I thought that the dataset could be augmented in two ways. If q1 is duplicated to q2 and q2 duplicated to q3 then create an entry with label duplicated for q1 and q3. The same way, if q1 is duplicated to q2 and q2 is not duplicated to q3, create an entry with label non duplicated for q1 and q3. This would allow to augment severely the dataset and thus make the model learn from many more created pairs of questions.

I also wanted to use the google API to translate all pairs of questions from english to spanish, french and german, and then translate back to english. This is very interesting because in this process the overall meaning of the question remains intact but words are slightly modified. It allows to augment the dataset and add more variations, which makes the model generalize better.

Finally, as there are a lot of very similar questions in the train and test set, I wanted to use a simple semi-supervised method call selfTraining. It iteratively trains on the train set and adds it at each iteration the most confident predictions of the test set with the predicted labels so as the train again and again until all samples of the test set are labeled.