M2 DATA SCIENCE

ADVANCED LEARNING FOR TEXT AND GRAPH DATA

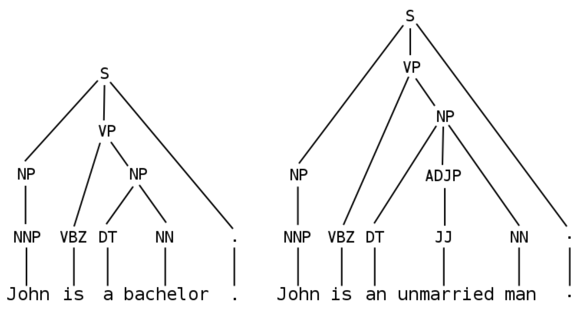
DATA CHALLENGE

# Can you predict whether two short texts have the same meaning?

Léa BRESSON
Eya KALBOUSSI

Dinh-Phong NGUYEN
Benoît ROBAGALIA

Team BKNR



## Contents

## 1 Introduction

The goal of this competition is to predict which of the provided pairs of questions contain two questions with the same meaning.

The ground truth is a set of labels supplied by human experts. This is inherently subjective, as the true meaning of sentences can not be known with certainty. Human labeling is a 'noisy' process, and different people would probably disagree. As a result, ground truth labels on this dataset should be taken as indications but not 100% accurate, and may include incorrect labeling.

## 2 Classical approach

### 2.1 Feature engineering

#### 2.1.1 TF-IDF

TF-IDF stands for "Term Frequency, Inverse Document Frequency". It's a way to score the importance of words in a document based on how frequently they appear across multiple documents. In our case, after some empirical testing, we chose not to use IDF and considered both unigrams and bigrams that appear at least three times in the whole set of questions. This served as a basis for further similarity measures.

#### 2.1.2 Cosine similarity

After representing the questions by vectors, we can get a good idea of their similarity by measuring the angle between the vectors. To make things even easier, by taking the cosine of this angle, we have a 0 to 1 value that is indicative of this similarity. The smaller the angle, the bigger the cosine value, and also the bigger the similarity.

#### 2.1.3 Levenshtein distance

The Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other [1]. To compute this distance, we used a very user-friendly library called *fuzzywuzzy*.

#### 2.1.4 Word match ratio

For each question, we calculated the number of words it has in common with its counterpart and then returned the ratio of shared words over total number of unique words in both questions. This feature had a significant impact on our performance; question pairs that had a ratio below 0.2 had all been labeled having a different meaning.

#### 2.1.5 Part-Of-Speech tagging

In this feature, we replaced each word by its grammatical category in the question. Then to compare the difference between the questions we considered several options; either to compute TF-IDF vectors and cosine similarity, or just the Levenshtein distance between the POS tagged sentences. The latter gave us the best performance of the two.

### 2.1.6 Redundancy

The data actually contains a lot of duplicates; there are a lot of questions that appear in both ”text_a” and ”text_b”, and that have the same counterparts. This feature captures the fact that the more a question has already appeared in the database, the more it is susceptible to have a duplicate.

### 2.1.7 Total number of words

We just added the number of words of each question. No real theoretical basis here, just pure empirical approach.

### 2.1.8 Difference in number of words

Here we calculated the absolute difference of number of words between both questions; we expect that the closer they are in meaning, the more similar the number of words.

### 2.1.9 Sentiment difference

This feature uses a very useful called *Vader* in *nltk*, that computes the level of positivity and negativity in a sentence. We used the squared difference of the compound score (taking into account positivity, neutrality and negativity) as a feature. If two questions have the same meaning, they are bound to express the same sentiment.

### 2.1.10 First word difference

As a very intuitive feature, we just used a binary indicator of the exact match of the first word of both questions; two questions that have the same meaning have more chance of beginning with the same interrogative word (e.g where, why, when, etc.).

### 2.1.11 Composition kernel in word embeddings space

For this feature, following a paper by Kim et al.[3], we implemented a version of the cosine similarity in the word embeddings space using pre-trained word vectors (300d Google word2vec). The sentence embedding was obtained using word vector addition, and similarity measures were calculated with those vectors (cosine similarity, euclidean distance, canberra distance).

### 2.1.12 Context similarity with graph of words

So far the features we built in a bag of words' approach don't really take into account the context of each word. Here, we create an undirected weighted graph of words where :

- Nodes are unique stemmed terms
- Edge between two nodes if they co-occur within a fixed-size sliding window (we chose 3 empirically).
- Edge weights match co-occurrence counts

Thus, the context of a word is represented by its corresponding neighborhood in the graph.

Then, we define a context similarity distance [4] as follows: $d(w_1, w_2) = \frac{|C(w_1) \cap C(w_2)|}{|C(w_1) \cup C(w_2)|}$ with $w_1$ and $w_2$ 2 words and $C(w_1)$, $C(w_2)$ their contexts respectively.

## 2.2 Model selection

### 2.2.1 Comparison of classifiers

We trained several classifiers (Random Forest, k-nearest neighbors, Logistic regression, multinomial Naive Bayes, AdaBoost, XGBoost) to predict whether or not two questions have the same meaning. To avoid overfitting, the parameters of each estimator are optimized using grid search with cross-validation.

### 2.2.2 XGBoost

In the end, the model that yielded the best performance was XGBoost, with a 70% features for each tree, a learning rate of 0.01 and a maximum tree depth of 5. As with other classifiers/regressors, cross-validation was performed with 20% of the training set as a validation set, and the model set to stop training if the loss calculated on the validation set did not improve, so as to avoid too much overfitting. Our final scores were the following: **public** = 0.18528, **private** = 0.18277; the two scores being pretty close, we can say there was no overfitting on the public part of the testing set. Interestingly, the feature that was the most "important" for the model was the graph-based measure of context similarity.

## 3 Deep learning

In this section we experimented with different deep learning architectures seen in class to get familiar with the Keras API. Parameter tuning was kept to a minimum due to lack of knowledge in the field.

## 3.1 Simple neural network

Using the previous features, we implemented a basic neural network with commonly used parameters: an input layer (300 nodes, ReLU activation), a single hidden layer (300 nodes, ReLU activation) and an output layer (1 node, sigmoid activation). Stochastic optimization was done with Adam and the chosen loss was naturally binary cross-entropy. We split the training set into a "training" set and a "validation" set (80%/20%), and kept the model that fared best on the validation set. The obtained score on the public leaderboard with this approach was 0.19751.

## 3.2 Convolutional neural network and word embeddings

This time we computed word vectors with the 300d GloVe 6B pre-trained embeddings, and obtained sentence embeddings using word vector addition. We fed them into a simple convolutional neural network with three convolutional layers and maxpooling, that we concatenated with a dense layer formed with the previously built features, and fed the whole to a last fully connected layer before outputing the probability via a sigmoid activation. The train/validation set split was 15% this time, and the obtained score on the public leaderboard was 0.18894 after 10 epochs and keeping the model that fared the best on the validation set.

## References

[1] Levenshtein, V.I. *Binary Codes Capable of Correcting Deletions, Insertions and Reversals* 1966: Soviet Physics Doklady, Vol. 10, p.707

[2] Hutto, C.J., Gilbert, E.E. *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text* 2014: Eighth International Conference on Weblogs and Social Media (ICWSM-14)

[3] Kim J., Rousseau F., Varzigiannis M. *Convolutional Sentence Kernel from Word Embeddings for Short Text Categorization* 2015: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 775–780

[4] Raeesi M. et al. *Trust Evaluation Using an Improved Context Similarity Measurement* 2014: International Journal of Business Information Systems Strategies (IJBISS) Volume 3, Number 1, February 2014