Machine Learning Course Final Project Report

Duplicate question identification

Denys Porplenko (porplenko@ucu.edu.ua), Volodymyr Maletskyi(v.maletskyi@ucu.edu.ua)

Problem Statement

Quora is a platform for the exchange of knowledge, which is visited monthly by about 100 million people. Users share knowledge by commenting / answering questions. There is a lot of topics, from cooking to mathematics. And these sites have one drawback that people ask similar (by intent) questions. For example,“What would a Trump presidency mean for current international master’s students on an F1 visa?” and “How will a Trump presidency affect the students presently in US or planning to study in US?” are duplicates. This forces people to continue the search for an answer (in search of the best), and the respondents answer the same questions several times (within the meaning of the intent).

To solve this problem, we used logistic regression, GBM and neural networks. For these models we made text pre-processing, used embedding, ti-idf, representation questions as graph (and compute some properties of graph) and others.

Data Preparation

Data source: https://www.kaggle.com/c/quora-question-pairs/data. The dataset contains pairs of questions and binary labels in a column ‘is_duplicate’. The dataset is human-labeled and contains some noise. Example:

1995	1996	How can I convert raw files to JPEG in photos in a MacBook?	How do you convert raw files to JPEG?	0
3506	3507	Can skipping increase your height?	What is the best and fastest way to increase your height?	1
5819	5820	Should I go to	Can I pursue	0

		school if I completely lost my voice?	masters in literature after engineering?	

We did text pre-processing on the data:

- Whitespace removal
- Words stemming
- Tags removal
- Replaced abbreviations with the full text
- Stopwords removal
- Replaced numbers (three, hundred etc) with digits
- Converted to lowercase

Embedding

Word2Vec is a model for representing words in a vector, which was invented by Tomas Mikolov (Google). Its main task is to present words in a vector space so that words that are close in meaning have a smaller vector distance than those that are far away.
This model uses the context of words. It uses two main methods: CBOW and Skip-gram. These two methods have opposite purposes: the CBOW architecture attempts to predict a word using a context, and Skip-gram to predict a context using a word.

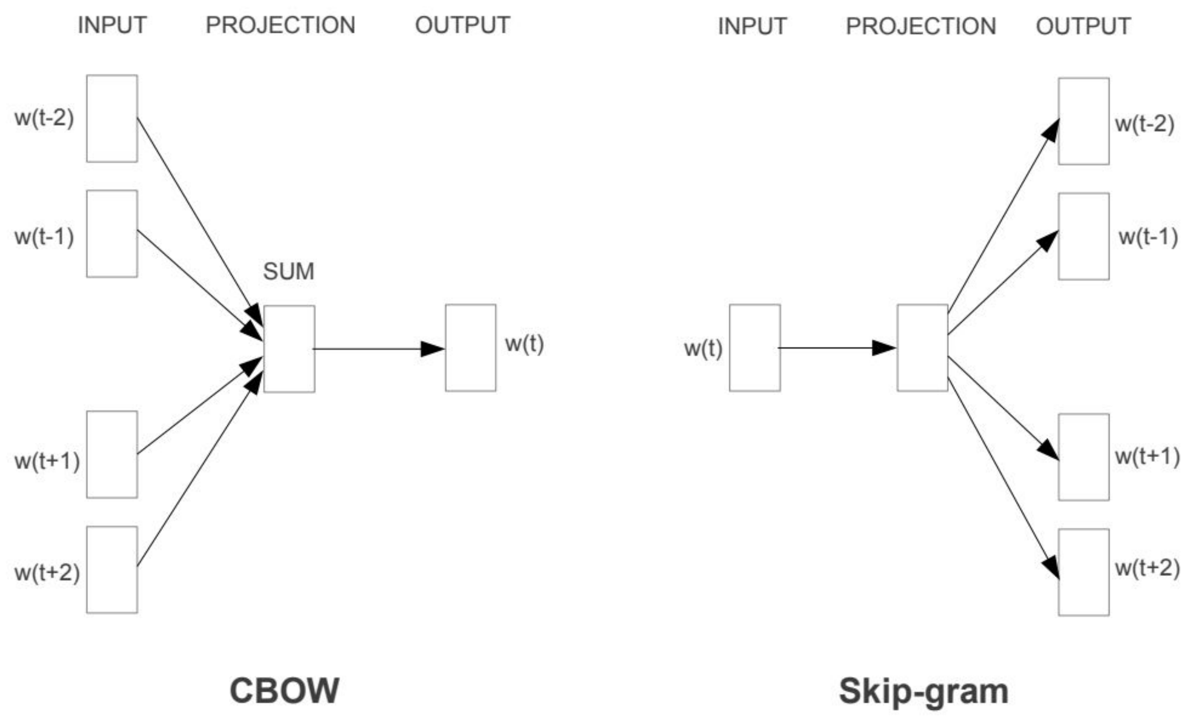


Figure 1: The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. Source: https://arxiv.org/pdf/1301.3781.pdf

For classification, we used a pretrained model GoogleNews-vectors-negative300.bin (pre-trained Google News corpus (3 billion running words) word vector model (3 million 300-dimension English word vectors)). We also used the intersect_word2vec_format method in Gensim to implement transfer learning. We trained the word2vec model on our dataset (questions), and then applied transfer learning from GoogleNews-vectors-negative300.bin. New words did not fall into our dictionary, but the existing ones (from the dataset with questions) changed their weights according to the pretrained model.

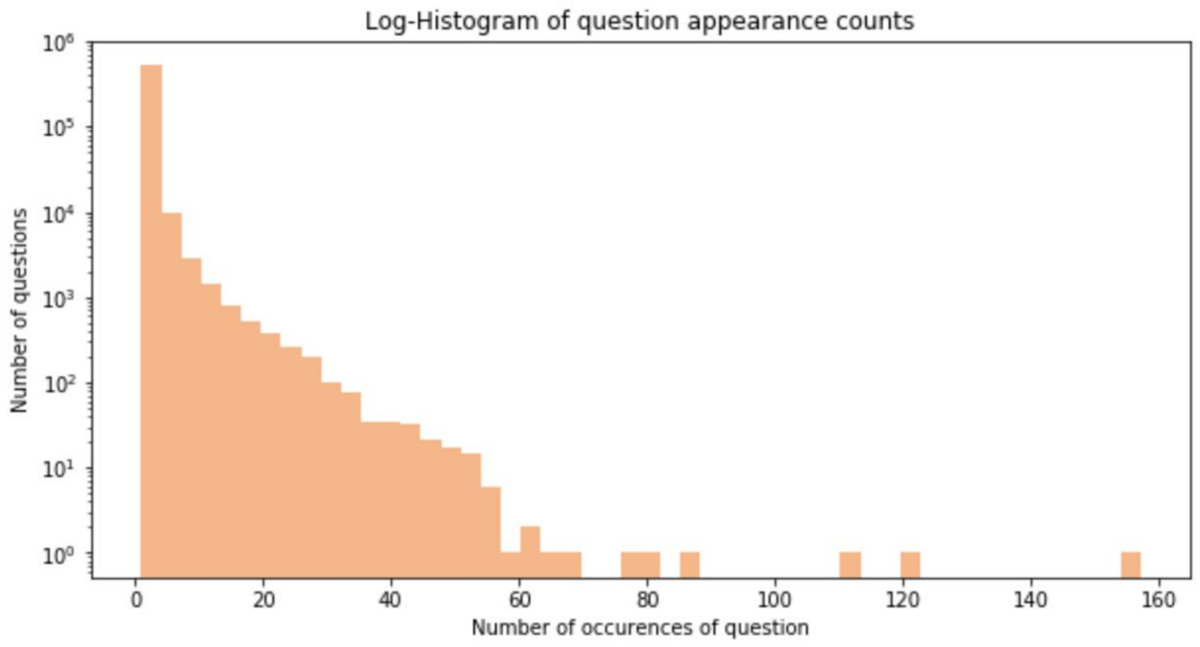
For these tasks, we used two powerful NLP libraries:

1. Gensim - is an open-source library implemented by Python / CPython for topic modeling and natural language processing. It was specially designed for working with large arrays of texts and also most of the calculations are done (in-memory). Implements: fastText, word2vec, doc2vec, latent semantic analysis (LSA, LSI, SVD), non-negative matrix factorization (NMF), latent Dirichlet allocation (LDA), tf-idf and random projections [1]
2. nltk - is a large platform that includes modules (implemented in Python), datasets. This platform is very useful in text processing for classification, tokenization, stemming, tagging, parsing, semantic reasoning. This platform is very widely used by linguists.

Exploratory data analysis

Total number of question pairs for training: 404287
Duplicate pairs: 36.92%
Total number of questions in the training data: 537929
Number of questions that appear multiple times: 111778

Log-Histogram of question appearance counts

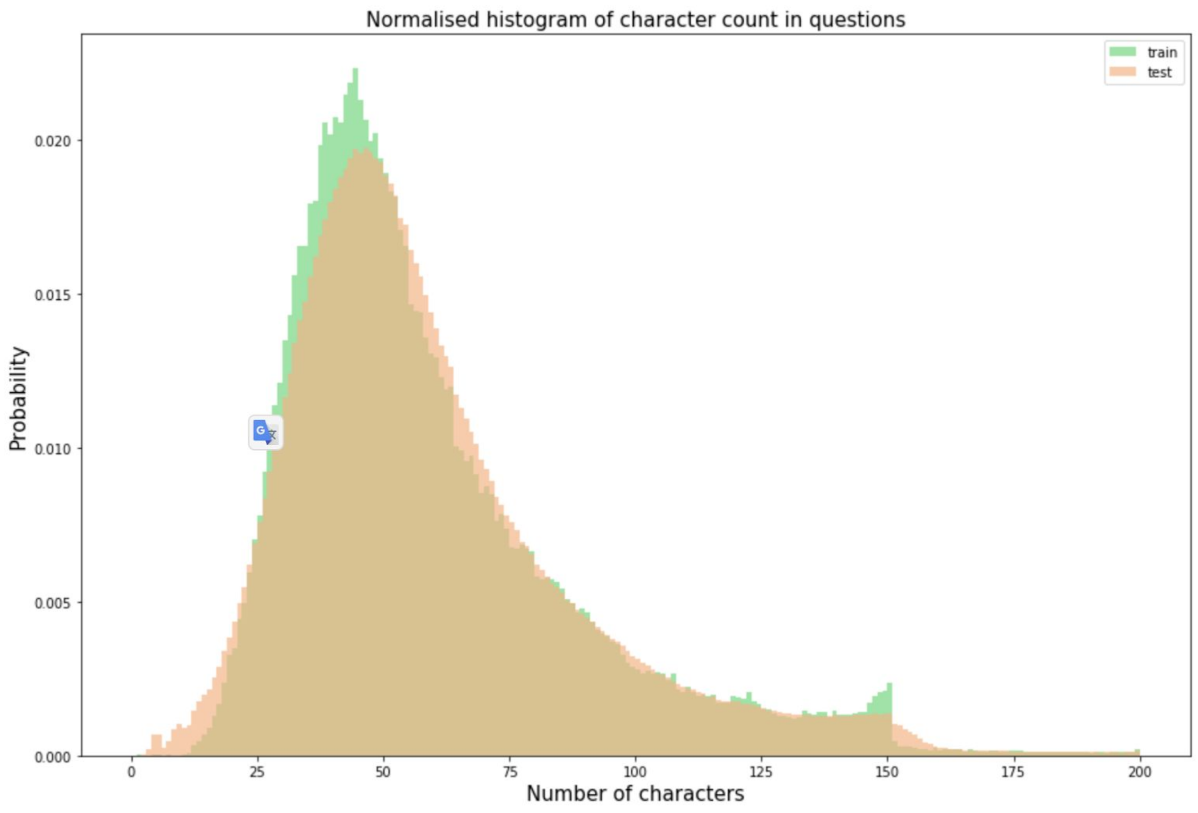


The biggest number of questions appear less than 10 times and almost all questions appear less than 50 times. There are several outliers that appear more than 100 times.

Count characters in questions

Mean train: 59.82
Std train 31.96
Max train: 1169.00

Mean test: 60.07
Std test: 31.62
Max test: 1176.00

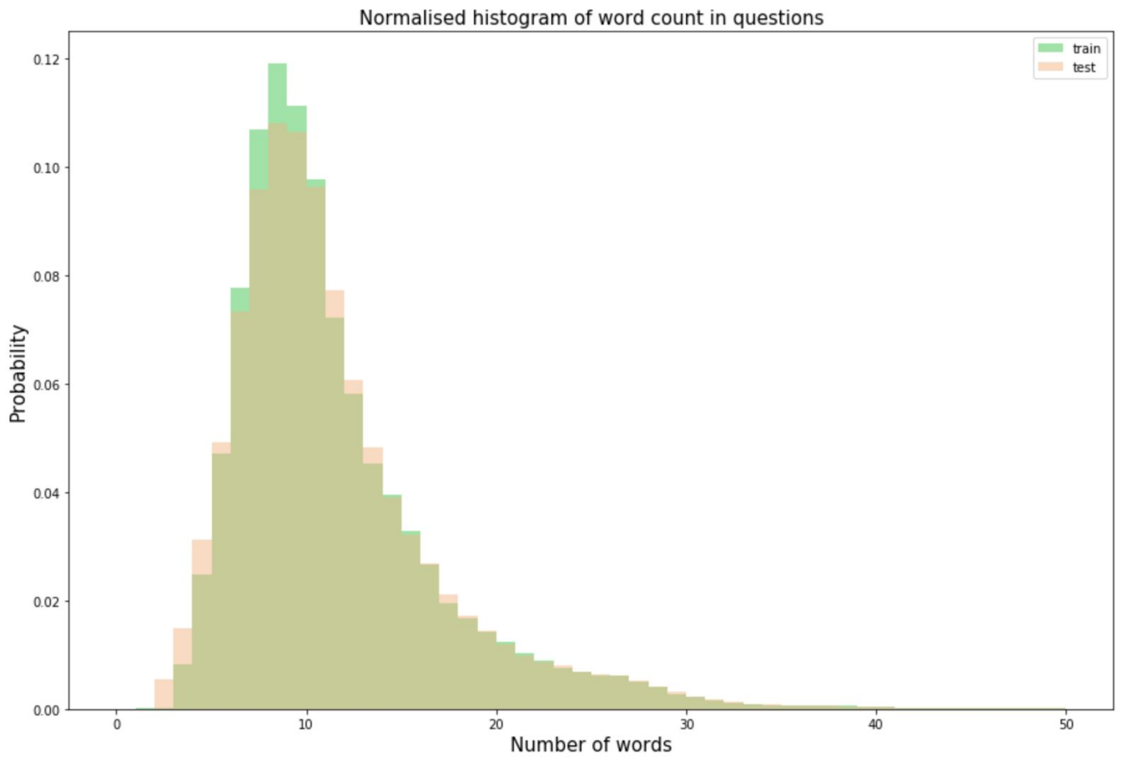


Most questions have length of between 20 and 150 characters with the mean around 60. We see the cut-off at 150, which is probably a limit on Quora.

Count words in questions

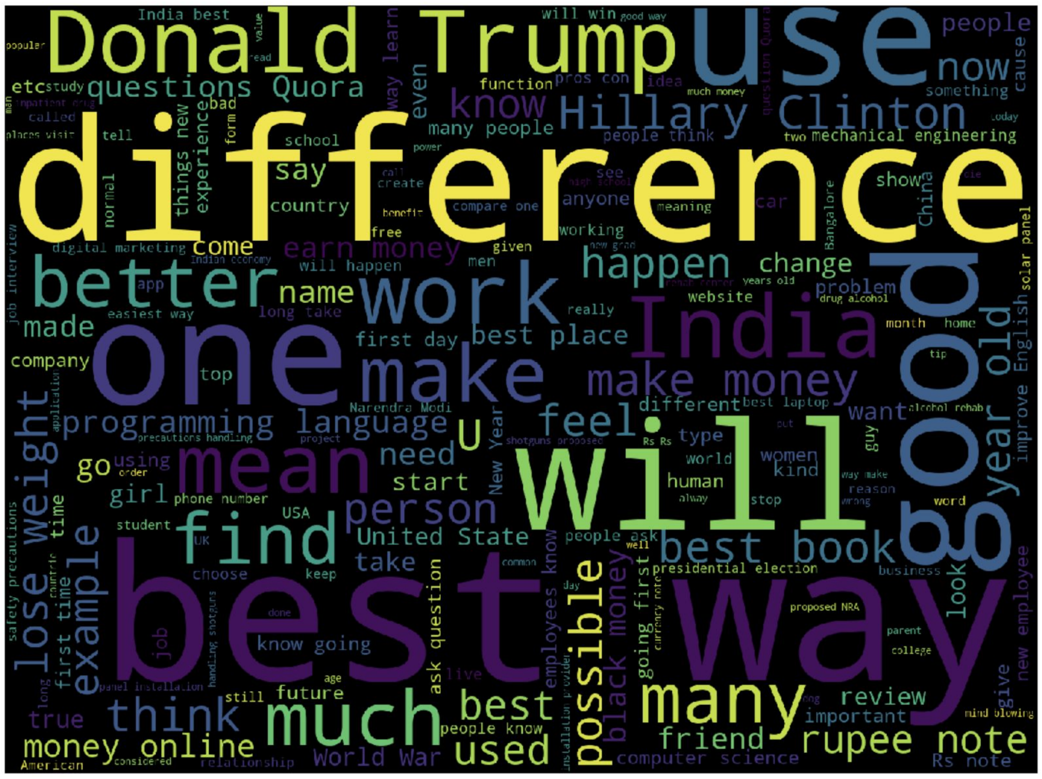
Mean train: 11.06
Std train 5.89
Max train: 237.00

Mean test: 11.02
Std test: 5.84
Max test: 238.00

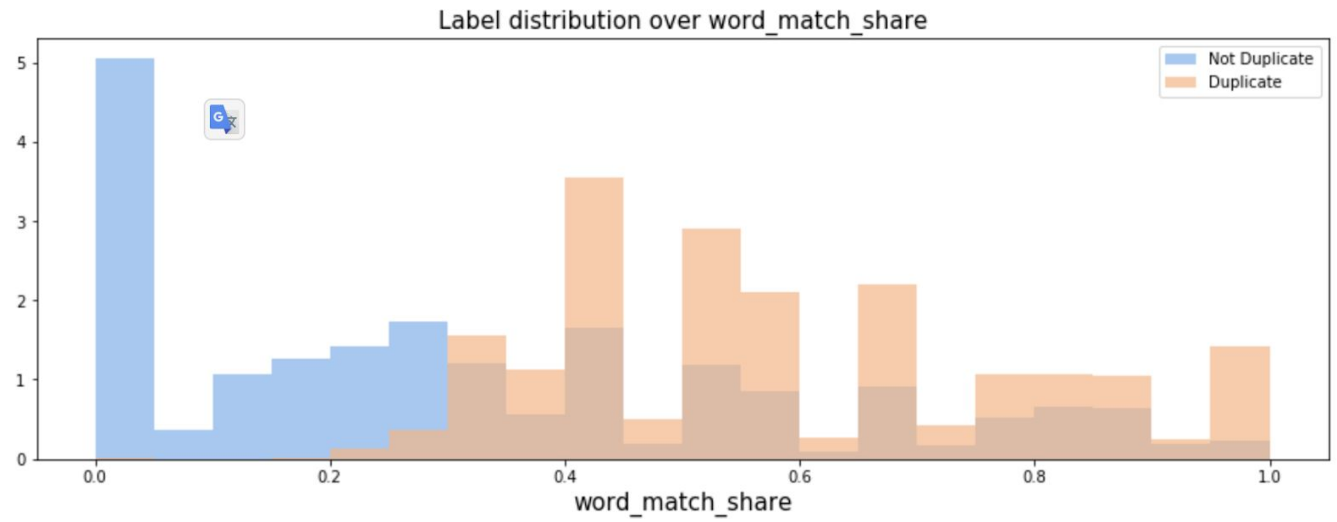


Distributions of word counts in train and test dataset are almost the same.

Words Cloud



Initial feature analysis: Shared words



Number of words shared between questions is quite a good predictor. Most non-duplicates have from zero to 0.2 shared words, whereas duplicates have all the way up to all words the same.

Feature engineering

We extracted three kinds of features: statistical features, natural language processing features and graph features.

Statistical features

We used python library FuzzyWuzzy to compute several kind of string distances between questions. We used the following metrics:

- Simple ratio between two strings: shows the edit distance between two strings - the number of edits required to get one string from the other (Levenshtein distance)
- Partial ratio: shows better result when the string are of different lengths. Returns Levenshtein distance between the shorter string and the best-matching substring of the second string.
- Token Sort Ratio: each string tokenized, tokens sorted, resulting strings compared with simple ratio method.
- Token Set Ratio: each string tokenized, token sorted into three groups - common for both strings and one group per each string. The ratio increases when intersection is bigger.
- Intersection ratio: the ratio of tokens common to both strings to the remaining.

Some quite trivial string measures:

- Absolute difference between the lengths of the strings
- Absolute difference between the numbers of question marks
- Absolute difference between the number of capital letters

NLP features

This set of fitch describes the relationship of two questions. Briefly: the **average weighted distance** between tokens in questions. Consider more details.

Distance. Each token (words) is represented as a vector. For example, $t1$ is a token (vector) from question $q1$ and $t2$ - this is the vector of token from question $q2$ (we compare tokens only from opposite questions). Each distance generates a separate feature. We used the following distances:

1. L1: $||t1 - t2||_1$
2. L2 (Euclidean distance): $||t1 - t2||_2$
3. Squared Euclidean Distance: $||t1 - t2||_2^2$
4. Minkowski Distance: $\sqrt[2]{\sum |t_1 - t_2|^p}$
5. Cityblock Distance: $\sum(t_1 - t_2)$
6. Canberra Distance: $\sum \frac{|t_1 - t_2|}{|t_1| - |t_2|}$
7. Chebyshev Distance: $\max |t_1 - t_2|$
8. Bray-Curtis Distance: $\frac{\sum |t_1 - t_2|}{\sum |t_1 + t_2|}$
9. Cosine Distance: $1 - \frac{t_1 t_2}{||t_1||_2 ||t_2||_2}$
10. Correlation Distance: $1 - \frac{(t_1 - \bar{t_1})(t_2 - \bar{t_2})}{||t_1 - \bar{t_1}||_2 ||t_2 - \bar{t_2}||_2}$

Weighted distance. Weights are the TF-IDF indicator for each word. The TF-IDF is a statistical measure used to assess the importance of a word in the context of a document that is part of a document collection or corpus. The weight of a word is proportional to the frequency of use of the word in the document and inversely proportional to the frequency of the word in all documents in the collection.

General formula:

$tf_{t,D} = \frac{n_t}{\sum_k n_k}$; n_t - count of token t in documents , $\sum_k n_k$ - count all tokens in the document

$idf_{t,D} = \log \frac{|D|}{|d : t_i \in d|}$; $|D| - countdocumentsinthecorpus$ $d : t_i \in d$ count documents with token k inside

$tf - idf_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_t}$

The weighted **average** distance between tokens in questions.
Term frequency – inverse document frequency for each token is weighted with different distances and the average value is on all tokens in the questions.

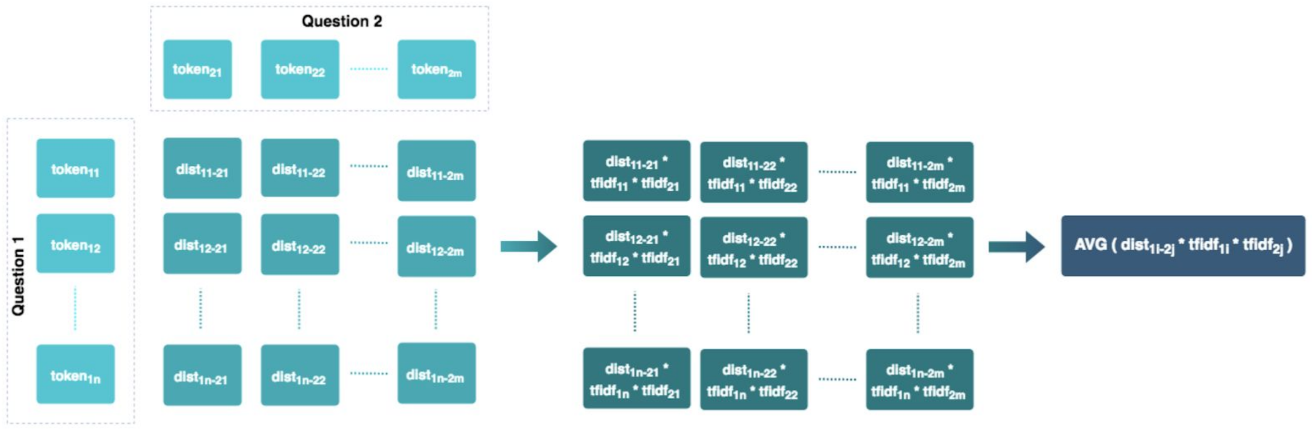
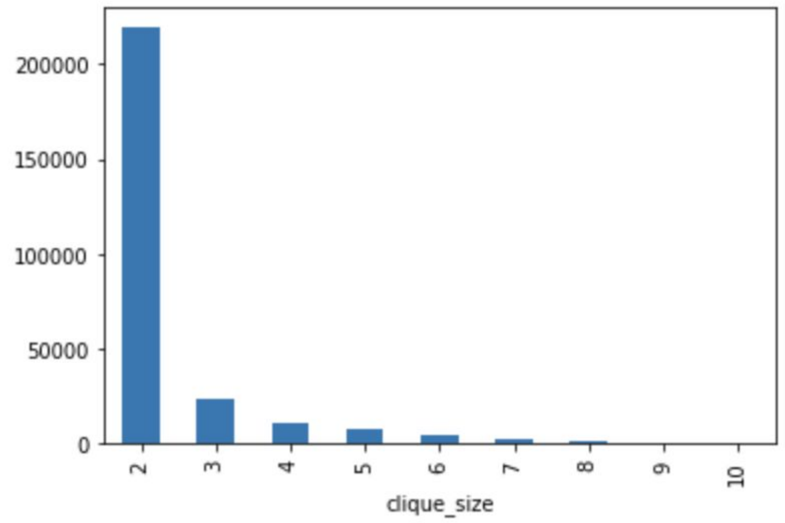

Question 2
$token_{21}$
$token_{22}$
$token_{2m}$
Question 1
$token_{11}$
$token_{12}$
$token_{1n}$
$dist_{11-21}$
$dist_{11-22}$
$dist_{11-2m}$
$dist_{12-21}$
$dist_{12-22}$
$dist_{12-2m}$
$dist_{1n-21}$
$dist_{1n-22}$
$dist_{1n-2m}$
$dist_{11-21} * tfidf_{11} * tfidf_{21}$
$dist_{11-22} * tfidf_{11} * tfidf_{22}$
$dist_{11-2m} * tfidf_{11} * tfidf_{2m}$
$dist_{12-21} * tfidf_{12} * tfidf_{21}$
$dist_{12-22} * tfidf_{12} * tfidf_{22}$
$dist_{12-2m} * tfidf_{12} * tfidf_{2m}$
$dist_{1n-21} * tfidf_{1n} * tfidf_{21}$
$dist_{1n-22} * tfidf_{1n} * tfidf_{22}$
$dist_{1n-2m} * tfidf_{1n} * tfidf_{2m}$
$AVG (dist_{1i-2j} * tfidf_{1i} * tfidf_{2j})$

Figure 3: Overview of feature set.

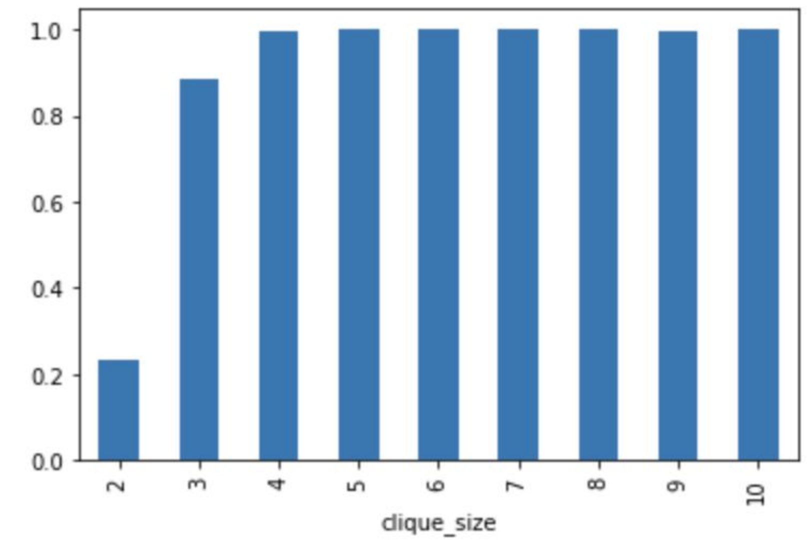
Graph features

We used python library networkx for building the graph where the questions are the edges, and are connected with the vertices if the dataset contains such pair of questions. We used this graph to extract clique sizes: a clique is a subset of vertices, such that each two vertices in the subset are adjacent. Cliques which have size of over 3 have about 60% of data labeled as duplicates. This is a so called “magic feature”, which is related to the structure of the dataset, not to the particular question pairs. It was discovered by Kaggle competition participants in [2].

Number of pairs grouped by clique size:



Ratio of duplicated questions grouped by clique size:



Models

Model 1: Logistic regression

Logistic regression is a model that analyzes the relationships between several independent variables and a dependent variable that can take a finite number of values. If the dependent variable can take only two values, then it is called a binary logistic regression. The current problem is related to binary logistic regression.

Cost function (which should be minimized):
$\min_{w,c} \frac{1}{2} w w^T + c \sum_{i=1}^{n} \log \left(e^{-y_i (X_i^T w + c)} + 1\right)$

Hyper parameters for this model:

1. C (Inverse of regularization strength)
2. Tolerance (Tolerance for stopping criteria)

To select the optimal model (optimal parameters) we used 5 fold cross-validation and Random search (sklearn.model_selection.RandomizedSearchCV)

Optimal parameters:
1. C (Inverse of regularization strength) : **10**
2. Tolerance (Tolerance for stopping criteria): **0.001**

Model 2: Gradient boosting (LightGBM)

Gradient boosting is an ensemble algorithm based on trees. Boosting is an iterative algorithm, when the next tree (model) is trained on data that was poorly identified by the previous model. The main idea of each tree is to improve the readings of the previous model.

There are several implementations of Gradient Boosting:
1. AdaBoost
2. XGBoost
3. CatBoost
4. LightGBM

I used the last, since it shows the highest learning speed for the current task (especially using the GPU)

Hyper parameters for this model:
1. Num_leaves (complexity of the tree model)
2. Reg_alpha (L1 regularization term on weights)
3. Min_data_in_leaf (prevent over-fitting in a leaf-wise tree)
4. Learning_rate (Boosting learning rate)

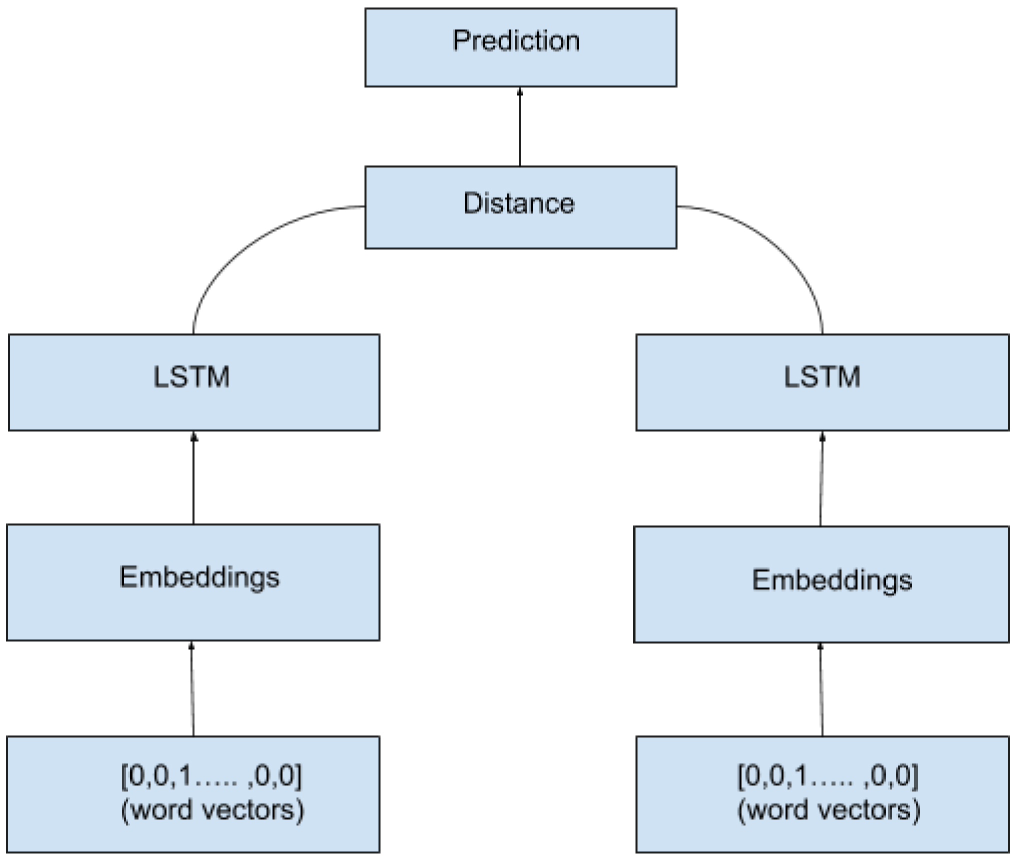
To select the optimal model (optimal parameters) we used 5 fold cross-validation and Random search (sklearn.model_selection.RandomizedSearchCV)

Optimal parameters:
1. Num_leaves: 31
2. Reg_alpha: 0.1
3. Min_data_in_leaf: 200
4. Learning_rate: 0.04

Model 3. Siamese Neural Network

Siamese NN is a type of neural network that is widely used for similarity learning in both computer vision and natural language processing. It consists of two identical neural networks, which in our case take as an input questions from each pair. Questions are represented as zero-padded sequences of word indices. These sequences are the input to the embedding layers of each of two networks, producing the matrix of embeddings, which represents the given question as a series of embeddings. Then each embedding matrix is fed to the corresponding LSTM - long short-term memory recurrent neural network. LSTM is a special type of recurrent neural networks designed specifically with the purpose of keeping long-term dependency across the recurrent neural net (also known as the vanishing gradient problem).

The structure of our neural network:



One LSTM cell has the following structure:

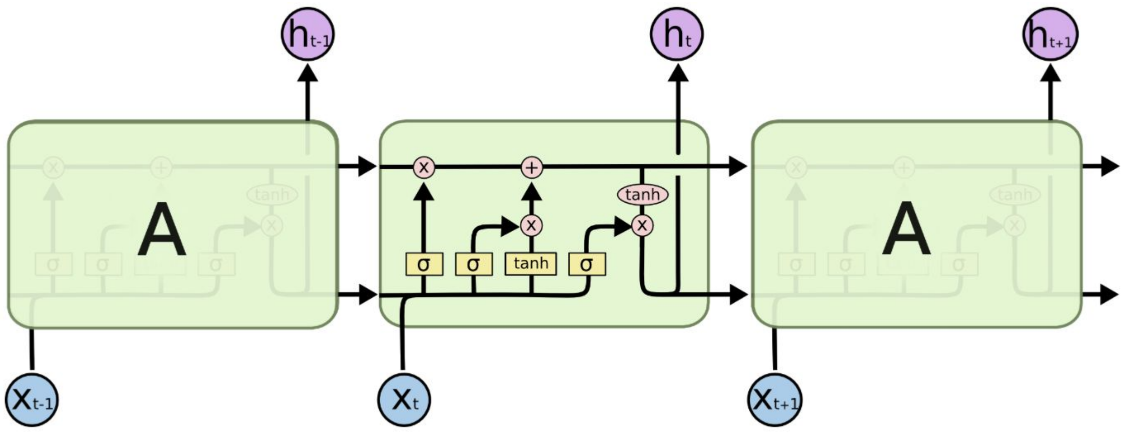


Image source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

We used the LSTM subtype called Manhattan LSTM, which is basically two LSTMs, used for measuring the distance between two sequence. As the similarity metrics we used the exponent of the Manhattan distance:

$$d = exp(-\sum_{i=0}^{\infty} \left| x_{left}^{(i)} - x_{right}^{(i)} \right|)$$

Evaluation

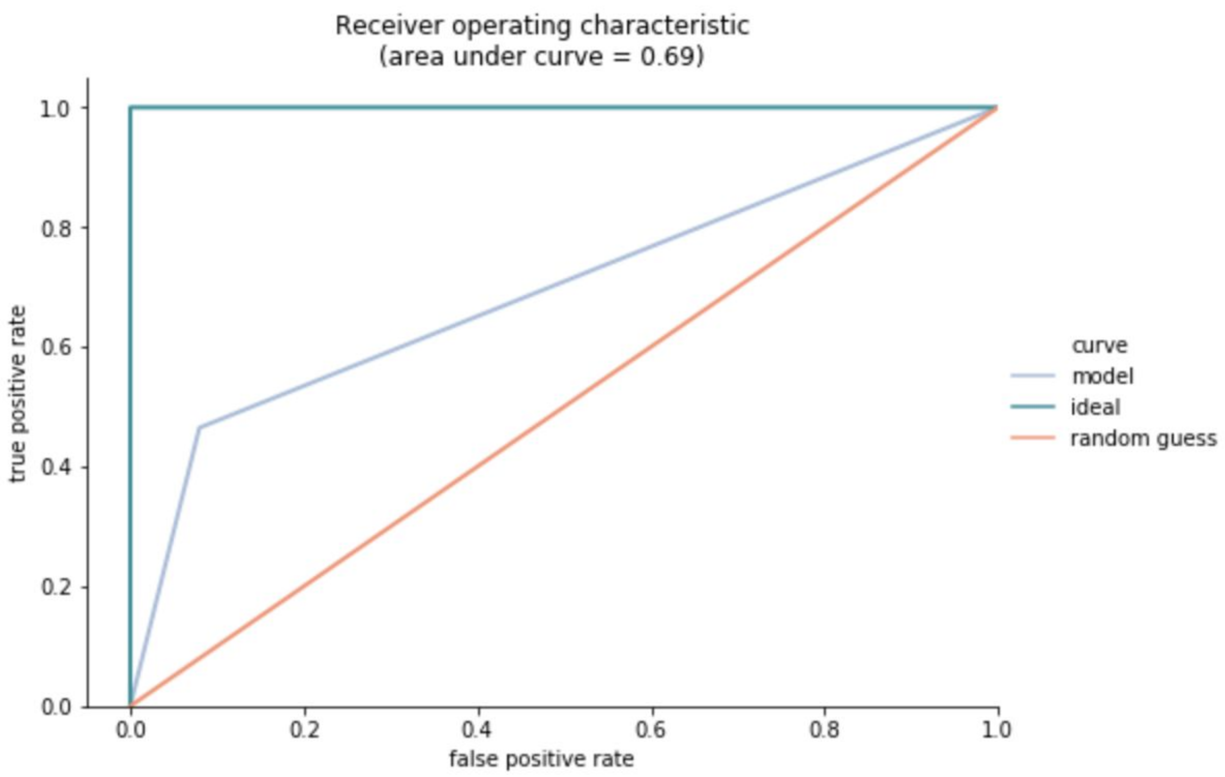
Model 1: Logistic regression

Prediction

	precision	recall	f1-score	support
not duplicate	0.75	0.92	0.82	83725
duplicate	0.77	0.46	0.58	49108
micro avg	0.75	0.75	0.75	132833
macro avg	0.76	0.69	0.70	132833
weighted avg	0.76	0.75	0.73	132833

Model Accuracy Score

accuracy score : 0.7519215857505288
precision score : 0.773859976267164
recall score : 0.46479188726887677
log loss : 0.4627793515862453



Model 2: Gradient boosting (LightGBM)

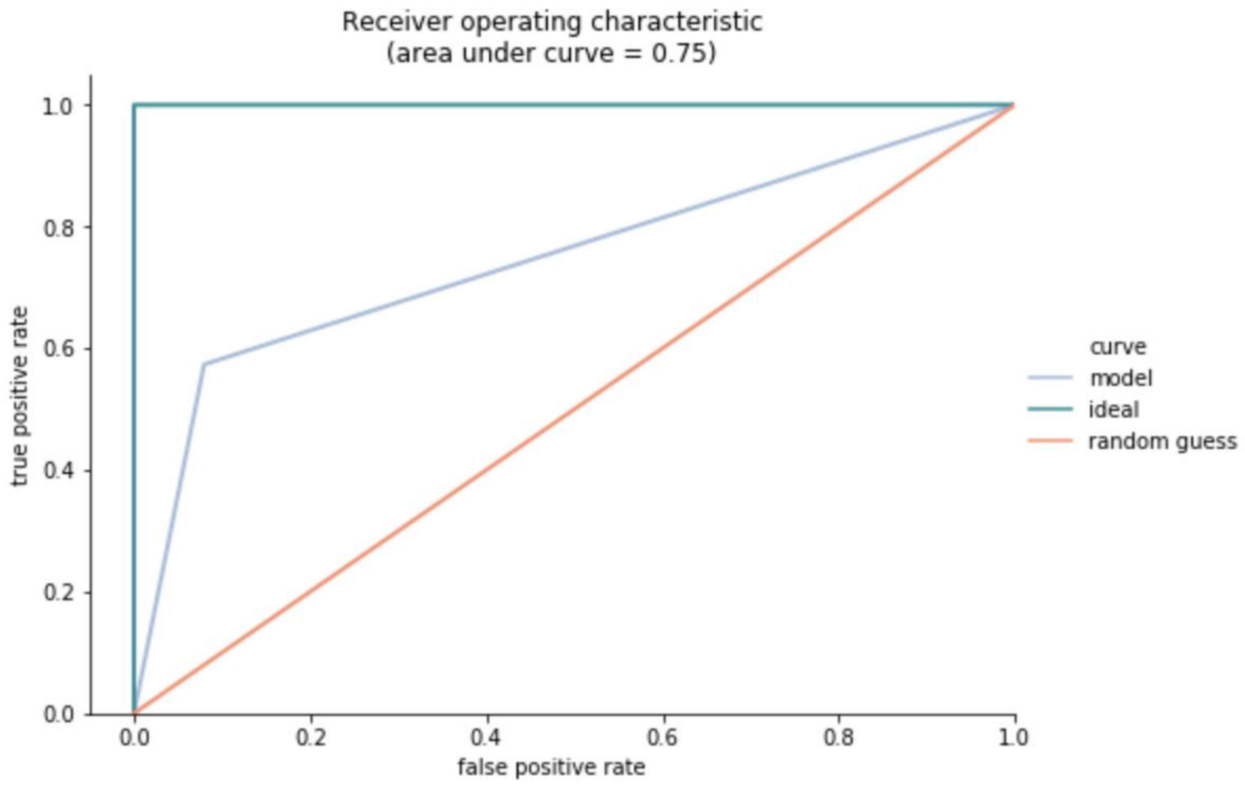
Prediction

	precision	recall	f1-score	support
not duplicate	0.75	0.92	0.82	83725
duplicate	0.77	0.46	0.58	49108
micro avg	0.75	0.75	0.75	132833
macro avg	0.76	0.69	0.70	132833
weighted avg	0.76	0.75	0.73	132833

Model Accuracy Score

accuracy score : 0.7932817899166623

```
precision score : 0.8121881579326854
recall score : 0.5734503543210883
log loss : 0.37493700898781807
```



Model 3: Siamese Neural Network

The model was trained during 6 epochs. The scores on the validation set were the following:

Log loss: 0.1539
Accuracy: 0.7854

Conclusion And Future Work

In this work we implemented several approaches to solving the short text similarity problem, using the Quora question pairs dataset. These approaches included logistic regression, gradient boosting using LightGBM and Siamese neural network. Both LightGBM and Siamese NN have shown quite successful results.

Due to the lack of computational resources, we were quite limited in the number and choice of the models. There are some options that proved themselves on similar tasks and are worth trying, e.g. ensembles, convolutional neural networks, random forests. Our plan for the future work include implementing stacking of the already implemented models (logistic regression, gradient boosting, Siamese Neural Net).

References:

1. https://en.wikipedia.org/wiki/Gensim
2. https://www.kaggle.com/tkm2261/my-magic-feature
3. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
4. Yukio Ichida, Alexandre & Meneguzzi, Felipe & Ruiz, Duncan. (2018). Measuring Semantic Similarity Between Sentences Using A Siamese Neural Network. 1-7. 10.1109/IJCNN.2018.8489433.

Previous work:

1. Detecting semantically identical questions pairs using Deep Learning https://github.com/vchennapalli/Quora-question-pairs
2. Kaggle Competition: Quora Question Pairs https://github.com/howardyclo/Kaggle-Quora-Question-Pairs
3. Quora Question Pairs [Zihan Chen, Hongbo Zhang, Xiaoji Zhang, Leqi Zhao] http://xiaojizhang.com/files/quora-question-pairs.pdf

Source:

1. https://github.com/DenisOgr/ucu-2019-ml-final-project (private repository)