Identifying Duplicate Questions

Machine Learning Engineer Nanodegree
Capstone Project

André Lukas Schorlemmer

July 17, 2017

1 Definition

1.1 Project Overview

Quora[1] is a question and answer homepage where people can ask questions which will then be answered by the community. Quora has a strong interest in only having one page for each unique question, because users who search for a specific topic or question would be able to access all the information in one place[2].

In this project classifiers are trained to detect short texts that the have same meaning. The dataset used for this is provided by Quora[2]. As of February 2017 Quora uses random forests in order to classify questions as duplicates[3]. In order to improve their internal matching algorithm they released the above mentioned dataset with question pairs and later hosted a kaggle competition[4].

1.2 Problem Statement

The Quora dataset consists of question pairs. As is mentioned above, the goal of this project is to develop an algorithm that is able to predict if these questions pairs are semantically equal. The problem is defined as a classification problem. The labels in the dataset are either 0 for non duplicate questions or 1 for duplicates.

In general it is useful to start with several basic preprocessing steps in order to find a solution for this problem. Tokenization is the first useful step. During this step the text is split into meaningful terms such as single words and symbols. Removing stop words is often helpful too. Stop words are very common words that can usually be removed without losing much information. Some examples for stop words are: a, the, and.

Vectorization is another preprocessing step that is often used in order to prepare text for the input of a classification algorithm. One common vectorization algorithm is called word2vec [1, 2, 3]. This algorithms will be used to turn words into vectors.

The raw text data as well as the tokenized text data will be used to calculate simple features that can be fed into a classification algorithm. Example of these features are the total number of characters in each question, the number of tokenized words, and the ratio of these two features. Furthermore, the created word vectors can be used to calculate distance measures between two questions.

[1] `https://www.quora.com/`
[2] `https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`
[3] `https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning`
[4] `https://www.kaggle.com/c/quora-question-pairs`

The preprocessed data will then be fed into classification algorithms. I am planning to compare three different algorithms. First a simple logistic regression classifier and boosted decision trees will be used. The input for these algorithms will be the newly created features mentioned in the paragraph above.

A long short term memory network (LSTM) will also be trained [4]. Different types of preprocessing will be tested together with the LSTM and the best type will be chosen based on its performance. The complete question text will be tokenized and vectorized as a first test and the second version will add additional preprocessing steps such as the removal of stop words. Two vectorization methods will be tested and compared. One of these methods will be the above mentioned word2vec model. The newly created word vectors will then be used as inputs for the LSTM network.

1.3 Metrics

Three metrics are used for the validation of the classification algorithms. The first metric is the the log loss:

$$L_{log} = -\frac{1}{N}\sum_{i=0}^{N-1} y_i \log(p) + (y_i - 1)\log(1-p)$$

where p_i is the predicted probability belonging to a class, y_i is the true label of the class and N the number of data points.

The log loss is used as the evaluation metric in the above mentioned kaggle competition. This is why this metric is also used in this project as the primary metric. The log loss encourages the usage of probabilities instead of simple class labels as predictors, because overconfident predictions are punished harshly. However, the log loss can be difficult to interpret. This is why the used classification algorithms are also compared using the accuracy and the receiver operation characteristic. The accuracy is defined the following way:

$$ACC = \frac{TP + TN}{N_{total}}$$

where TP is the number of true positives, TN is the number of true negatives and N_{total} is the total number of data points in the dataset.

The receiver operation characteristic (ROC plot) is obtained by scanning through the prediction threshold and then plotting the true positive rate versus the false positive rate for the corresponding threshold values. More details about ROC plots can for example be found on Wikipedia[5].

2 Analysis

2.1 Data Exploration

The dataset used in this project is available as a csv file on kaggle[6]. It is also available at other locations[7][8]. The dataset is subject to the Quora terms of usage which allow non commercial use[9]. The zipped dataset has a size of 21 MB.

[5] https://en.wikipedia.org/wiki/Receiver_operating_characteristic
[6] https://www.kaggle.com/quora/question-pairs-dataset
[7] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
[8] https://data.world/socialmediadata/quora-question-pairs
[9] https://www.quora.com/about/tos

The data consists of six columns. Each data point has its own identifier (id) and each question has its own unique identifier (qid1, qid2). The most important columns are the two questions that were posted on Quora (question1 and question2) and the column is_duplicate shows if these questions are duplicates. This ground truth may contain some noise, since semantically equality can not always be defined clearly. Some example questions are shown in table 1.

Id	qid1	qid2	question1	question2	is_duplicate
7	15	16	How can I be a good geologist?	What should I do to be a great geologist?	1
22	45	46	What are the questions should not ask on Quora?	Which question should I ask on Quora?	0
94	189	190	What is abstract expressionism in painting?	What are some of the major influences of abstract expressionism?	0
104	209	210	Can I recover my email if I forgot the password?	What should I do if I forgot my email password?	1
1918	3837	3838	How can I speed up my Internet connectionn?	How can I speed up my Internet connection?	1
398564	3838	778740	Why do we sleep?	Why do we sleep during the night?	0
403048	787314	3838	Why do human need sleep?	Why do we sleep?	1

Table 1: Examples of the text data that are used in this project. Each question has a unique id because some questions appear multiple times in the dataset.

The dataset consists of 404351 question pairs in total. 149306 of the pairs are tagged as duplicates which is about 37% of the whole dataset. The questions are in general relatively short, with a mean count of approximately 11 words and a maximum of 237 words[10]

The questions in the dataset are not unique and some of the questions appear several times in the dataset as it is shown in table 1. The dataset is noisy as is demonstrated in the rows with the Ids 1918 and 403048. One of the question ids is mislabelled and the data contains questions with typos and grammatical mistakes.

The problem is very similar to a competition that was launched by kaggle on March 16[11]. The training data for the kaggle competition is very similar. It has the same size and a visual inspection reveals many question pairs that are equal. However, the kaggle dataset consists only of 404290 question pairs. It seems that some questions have been removed. Kaggle also provides a test set that has a size of 112 MB. The test set is not labelled and it is enriched with fake questions that have not been posted on Quora in order to prevent cheating. No information could be obtained on the cause of the size reduction of the kaggle data. This project only uses the dataset which was published on January 24 by Quora and not the data of the kaggle competition.

As was mentioned, text data needs to be preprocessed and turned into features before classification algorithms can be used. The goal of feature creation is to capture the similarity between

[10]Words were extracted with a str.split() and no further preprocessing steps were applied.

[11]`https://www.kaggle.com/c/quora-question-pairs`

the two questions in different ways. For example: similar questions could be more likely to have a similar length. By just counting the number of characters in each question a simple feature is created that is capable of capturing this similarity. Results of the counting are shown in in figure 1. Only the training set is shown (for more details see section 3.1). It is visible that the duplicate questions have a smaller length difference in total. However, many non duplicate question still have a similar length, which is to be expected for a complex problem like this one. However, it should be possible to classify the question with a decent performance if enough features can be found that exploit the similarity between duplicate questions in different unique ways.

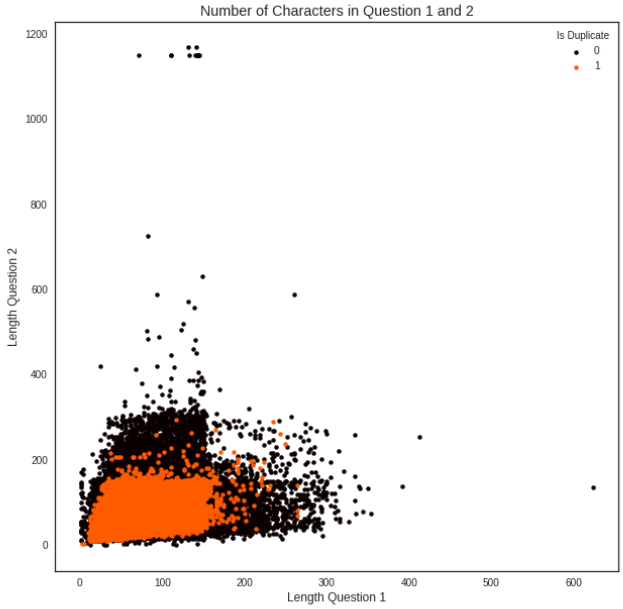


Figure 1: Number of characters of question 1 and question 2 shown for questions that are duplicates (orange) and non-duplicates (black). Only data points from the training set are shown.

2.2 Algorithms and Techniques

As mentioned in section 1.2 three different classification algorithms are used and compared to find the best one. All classifiers will predict a number between 0 and 1, which can be interpreted as the probability that the corresponding question pair is a duplicate. Questions are considered duplicated if the probability is above 50%.

The logistic regression classifier is used as a simple algorithm to get started with the preprocessed data only. The data preprocessing is described in more detail in section 3.1. Only the default parameters are used.

The second class of algorithms that is used are boosted decision trees. Boosted decision trees have been used in many kaggle competitions and are often part of the winning solution[12]. These

[12]http://blog.kaggle.com/2017/01/05/your-year-on-kaggle-most-memorable-community-stats-from-2016/

classifiers should be tuned in order to improve the performance. The tuning of boosted decision trees can be difficult and time consuming because they have a large number of parameters. Important parameters that are often tuned are described in the following. The description of the parameters shown below are taken from the XGBoost documentation[13]:

- n_estimators: Number of boosted trees to fit.
- max_depth: Maximum tree depth for base learners.
- min_child_weight: Minimum sum of instance weight(hessian) needed in a child.
- subsample: Subsample ratio of the training instance.
- colsample_bytree: Subsample ratio of columns when constructing each tree.
- learning_rate: Boosting learning rate.

The implementation of the hyperparameter tuning is described in detail in section 3.3. The final model that is studied in this project is a LSTM network. The layout of the network is based on a kaggle kernel[14]. Figure 2 shows a schematic overview of the network. In its final version after hyperparameter tuning the neural network has the following layout.

The preprocessed data is fed into an embedding layer which forwards the words into one LSTM layer with 230 nodes. The weights of the LSTM layer are equal for both questions. The activation for this layer is a hyperbolic tangent and the activation for the recurrent step is a hard sigmoid function. The output for question 1 and 2 is concatenated. Batch normalization as well as dropout layers are then used to speed up learning and in order to mitigate the effects of overfitting. The output is fed into a dense layer with 128 nodes. Rectified linear units (ReLus) are used as activation functions for this step. Afterwards another batch normalization layer and another dropout layer is added. Finally the data is routed into a dense layer with one node and a sigmoid as its activation function in order to create a prediction. The LSTM also has many parameters that are used to tune and improve the performance of the model:

- The number of epochs defines the number of training steps.
- The batch size defines the number of data points that are taken into account during a training step.
- The maximum sequence length defines the maximum number of words or tokens that are taken into account per question.
- The number of LSTM units/nodes.
- The number of dense units/nodes.
- Dropout probability of the LSTM layer.
- Dropout probability for the dense layers.

[13] `https://xgboost.readthedocs.io/en/latest/python/python_api.html`
[14] `https://www.kaggle.com/lystdo/lstm-with-word2vec-embeddings`

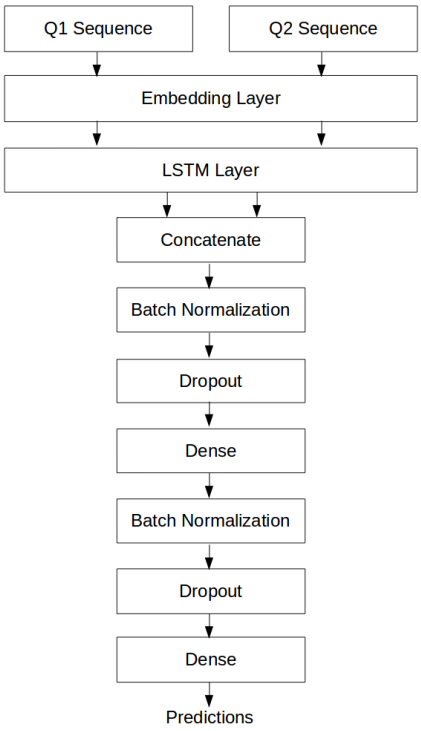


Figure 2: Layout of the LSTM network that is used to classify the data.

2.3 Benchmark

The benchmark model is based on counting the words that are equal. It is the same benchmark that is used in the mentioned Quora kaggle competition. The code to produce the benchmark is based on a published kaggle kernel[15].

The questions are tokenized and stop words are removed using the nltk corpus [6]. The prediction p is then calculated by averaging over the two fractions of shared words:

$$p = \frac{1}{2}\left(\frac{n_{shared}}{n_{q1}} + \frac{n_{shared}}{n_{q2}}\right)$$

The number of words that both questions share is called n_{shared}. Question 1 consists of n_{q1} words and question 2 of n_{q2} words.

3 Methodology

3.1 Data Preprocessing

First the data is split into a training, a validation, and a test set. The test size is 10% of the total dataset size. 20% of the 90% that are remaining are used as the validation set and the rest of the data is used as the training set.

As is mentioned the preprocessing steps differ depending on the classification algorithm. The preprocessing for the logistic regression and the boosted decision trees consists of the following

[15] `https://www.kaggle.com/cgrimal/quora-question-pairs/words-in-common-benchmark/code`

steps.

Both questions are tokenized. All of the words are then lemmatized. As a next step stop words are removed using the nltk stoplist. Finally punctuation symbols are removed. Several new features are derived from the text data. An excellent blog post was used as a guideline for the feature creation [16]. The new features can be split into two main categories: Features that are derived directly from the text data and features that measure similarities between word vectors. The directly derived features are:

1. q1_length: Number of characters in question 1 (no preprocessing)
2. q2_length: Number of characters in question 2 (no preprocessing)
3. diff_length: Difference of q1_length and q2_length
4. q1_n_words: Number of words in question 1 after application of preprocessing
5. q2_n_words: Number of words in question 2 after application of preprocessing
6. q1_len_word_ratio: Ratio between q1_length and q1_n_words
7. q2_len_word_ratio: Ratio between q2_length and q2_n_words
8. word_share: Output of the benchmark model used as new input

A pretrained word2vec model[17] is used to turn the preprocessed words into 300 dimensional vectors. The mean vector is then calculated from the vectors of each question. The similarity between the two remaining vectors for each data point is then measured by the following features:

1. Cosine distance between vector 1 (mean vector of question 1) and vector 2 (mean vector of question 2)
2. Cityblock distance between vector 1 and vector 2
3. Jaccard distance between vector 1 and vector 2
4. Canberra distance between vector 1 and vector 2
5. Minkowski distance between vector 1 and vector 2
6. Euclidean distance between vector 1 and vector 2
7. Braycurtis distance between vector 1 and vector 2
8. Skewness of vector 1
9. Skewness of vector 2
10. Kurtosis of vector 1
11. Kurtosis of vector 2

[16] https://www.linkedin.com/pulse/duplicate-quora-question-abhishek-thakur
[17] https://code.google.com/archive/p/word2vec/

As is explained in section 1.2 different versions of preprocessing are tested on the LSTM network in order to find the best one. The first version keeps the preprocessing steps for the LSTM network to a minimal amount. The whole questions are tokenized. During this process all words are lowered and symbols such as punctuation are removed. The list of symbols that are removed is: `"#$%&()*+,-./:;<=>?@[\\]^_‘{|}~\t\n`. The mentioned word2vec model is used to create a word embedding and the data is then finally fed into the LSTM network that is described above. The second version uses the above described tokenized data as inputs for the word2vec model instead of the full questions. Furthermore, glove embeddings [5] are also tested as a replacement for the word2vec model. All of the refinement steps are described in detail in section 3.3.

3.2 Implementation

The project uses several files and folders. The algorithms were implemented and trained using Jupyter Notebooks. Functions that were used several times are split up and are located in the capstone_project folder. An overview of the project organization is given in table 2. More detailed instructions are given in the README file.

File/Folder	Description
data/	The original data
notebooks/	Jupyter notebooks
output/	Preprocessed data, models etc.
tests/	Unit tests
capstone_project/	Project source code
environment.yml	conda definition file
LICENSE	
README.md	
setup.py	

Table 2: Structure of the capstone project.

The project is implemented using python 2.7. Some of the most important packages are mentioned in the following, while the complete list of used packages is found in the environment.yml file. The three classification algorithms use different packages. The scikit-learn implementation[18] is used for the Logistic Regression Classifier, xgboost[19] is used to implement the boosted decision trees. Keras[20] is used for the implementation of the LSTM network.

Several problems had to be solved during the development. Originally it was planned to use one pipeline to implement the preprocessing and classification for the logisitic regression classifier and the boosted decision trees. However, processing the whole text corpus every time proved to be very time consuming. This is why the analysis of the data was split up into several jupyter notebooks. The output of every step is saved on disk. The first notebook splits the dataset and tokenizes the questions, the second notebook creates the new features. Classifiers are finally applied in separated notebooks.

The implementation of the LSTM network also proved to be challenging. The validation loss of the LSTM network stopped improving after about 5 to 10 epochs and remained at a high value, while the training performance continued to decrease. Initially it was assumed that the

[18] `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`
[19] `https://xgboost.readthedocs.io/en/latest/`
[20] `https://keras.io/`

network was already overfitting and several different network parameters were used to train the model. Later it was found that the splitting of the second epoch validation set did not work correctly (for more details on the epoch validation read section 3.3). The code used for the splitting was deleted and reimplemented and the network started training correctly afterwards.

The LSTM network also started to predict a few duplicates with such a high confidence that python automatically rounded the predicted float to one. The scikit implementation of the log loss returns a nan value if any of the prediction is equal to one. Therefore a quick fix was added. All confident predictions are capped at a maximum value of 0.9999999.

3.3 Refinement

The logistic regression classifier is not optimized as mentioned. The boosted decision tree classifier is tuned using random search and cross fold validation. The learning objective is always set to "binary:logistic" in order to minimize the cross-entropy. A blog post was used as a guideline for the parameter tuning of the xgboost classifier[21]. The classifier is first trained with a high learning rate of 0.2 and the initial parameters[22] which are shown in table 3.

Parameter	Value
n_estimators	1000
max_depth	10
min_child_weight	1
gamma	0
subsample	0.8
colsample_bytree	0.8
reg_alpha	0

Table 3: Initial parameters of the XGBoost Classifier.

The best number of estimators is then found by using early stopping rounds. The classifier is trained using cross fold validation on the training data and the performance is evaluated at every training step. The training is stopped as soon as the mean loss of the validation folds has not improved for 50 steps. Afterwards the parameters are tuned using random search and cross fold validation. After the best parameters are found, the learning rate is reduced to 0.01 and the best number of estimators is then again obtained the same way. The final parameter values are shown in table 4.

The mean of the log loss using five fold cross fold validation for the initial model is found to be 0.4185, while the log loss of the tuned model is 0.3980.

The LSTM model on the other hand is not trained using cross fold validation due to computational constraints. The training data is instead split into another training subset and a second validation set that is used to evaluate the performance at every training step. The size of the second validation set is 10% of the training dataset. The second validation set is referred to as the epoch validation set from now on in order to avoid confusion with the first validation set that is used to compare different classification algorithms.

The initial parameters of the LSTM network are reported in table 5, where MSL stands for maximum sequence length. The minimum log loss of the epoch validation set is also shown in

[21] https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[22] Detailed description of parameters: `https://xgboost.readthedocs.io/en/latest/parameter.html`

Parameter	Value
n_estimators	10598
max_depth	10
min_child_weight	1
gamma	0.37
subsample	0.854
colsample_bytree	0.722
reg_alpha	1.529

Table 4: Final parameters of the XGBoost Classifier.

the table. The preprocessing of the first model uses the whole question as input and the glove model vector embeddings as described in section 3.1 .

Batch size	MSL	Num LSTM	Num Dense	Dropout LSTM	Dropout dense	Loss
2048	30	230	128	0.3	0.3	0.3801

Table 5: Initial parameters of the LSTM network using glove embeddings as inputs.

The initial number of estimators is set to 200. Again early stopping rounds are used to find a good number of estimators. However, since the training of the LSTM is slow only 3 rounds are used as patience before the training is stopped. Results using the complete questions and the word2vec model are shown in table 6.

Batch size	MSL	Num LSTM	Num Dense	Dropout LSTM	Dropout dense	Loss
2048	30	230	128	0.3	0.3	0.3740
2048	30	256	128	0.3	0.3	0.3781
512	30	230	128	0.3	0.3	0.3866
256	30	230	128	0.3	0.3	0.3870
2048	50	230	128	0.3	0.3	0.3718
2048	50	230	128	0.3	0.3	0.3775
1024	100	230	128	0.3	0.3	0.3843
2048	50	230	128	0.4	0.4	0.3707
2048	30	230	128	0.35	0.35	0.3750
2048	30	230	128	0.35	0.35	0.3700
2048	30	230	128, 64	0.3	0.3	0.3746

Table 6: Tested parameters of the LSTM network using the word2vec embeddings as inputs. The last row shows an implementation of the LSTM with two dense layers.

The raw data input and the word2vec implementation work well and have been tested more thoroughly. The difference in performance is in general very low. All models have a minimum loss between 0.37 and 0.39. The results are also not always reproducible as it is visible in table 6, rows (5, 6) and (9, 10). Results like these can be expected since the loss function is highly non convex and the patience is set to a low value. It seems that the backpropagation algorithm ends up in different local minima in a random fashion, while the tested changes in parameters do not seem to be playing a major role in model performance. Thus the best performance depends to a certain degree on luck. The tokenized text data was also tested. The results for the tokenized data are reported in table 7.

Batch size	MSL	Num LSTM	Num Dense	Dropout LSTM	Dropout dense	Loss
2048	30	230	128	0.3	0.3	0.3896

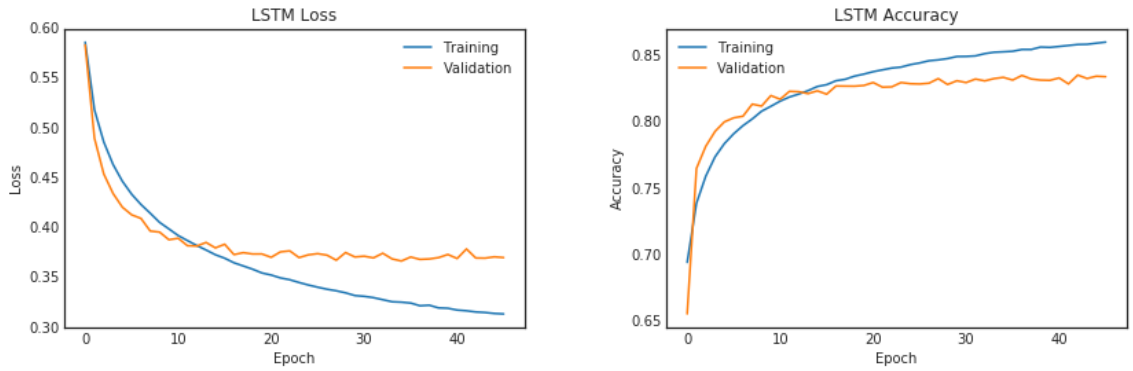
Table 7: Tested parameters of the LSTM network using the word2vec embeddings and the tokenized inputs.

The final parameters are shown in table 8. The patience is set to 10 in this case in order to increase the chance of finding a better local minimum. These parameters were chosen because they seem to be a good trade-off between training time and performance. A similar loss was achieved with a higher dropout but the training time increased at the same time.

The performance development during training of the final LSTM network is illustrated in figure 3. A the beginning the loss is very high and the accuracy is low for the training set as well as the epoch validation set. The loss decreases quickly and for a while the validation loss is smaller than the training loss. This could be caused by the dropout layers, because dropout is only active during training and the full network is used for predictions. Finally the performance on the validation set reaches a plateau, while the training loss continues to decreases and the network starts to overfit.

Batch size	MSL	Num LSTM	Num Dense	Dropout LSTM	Dropout dense	Loss
2048	30	230	128	0.35	0.35	0.3662

Table 8: Final parameters of the LSTM network using the word2vec embeddings as inputs. Patience is set to 10.



(a) (b)

Figure 3: Development of the loss (a) and accuracy (b) of the LSTM network using the final parameters on the training data subset.

4 Results

The performance of the tuned classifiers is compared using the validation set in order to find the best model. The log loss, the accuracy, as well as the ROC curve are calculated. The log loss and the accuracy of the three classifiers and of the benchmark model are reported in table 9.

Model	Log Loss	Accuracy
Benchmark Model	0.7603	0.69
Logistic Regression	0.5244	0.68
XGBoost Classifier	0.3903	0.79
LSTM network	0.3720	0.83

Table 9: Performances of tested classifiers. Only predictions done on the validation set are shown.

The benchmark model and the logistic regression show a very similar performance. It is interesting that the logistic regression has a lower log loss than the benchmark model and is thus able to classify the questions with a higher confidence. The accuracy however is very similar and the logistic regression is not able to improve the total amount of correctly predicted question pairs. The XGBoost Classifier as well as the LSTM network show a better performance. The log loss and the accuracy are improved compared to the benchmark model and the LSTM network shows the best performance overall. ROC curves for these models are shown in figure 4 to further validate the model performance.

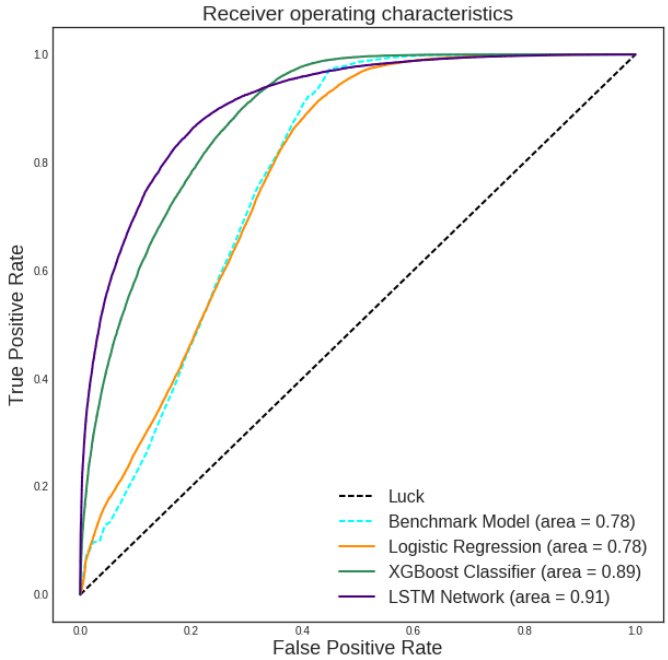


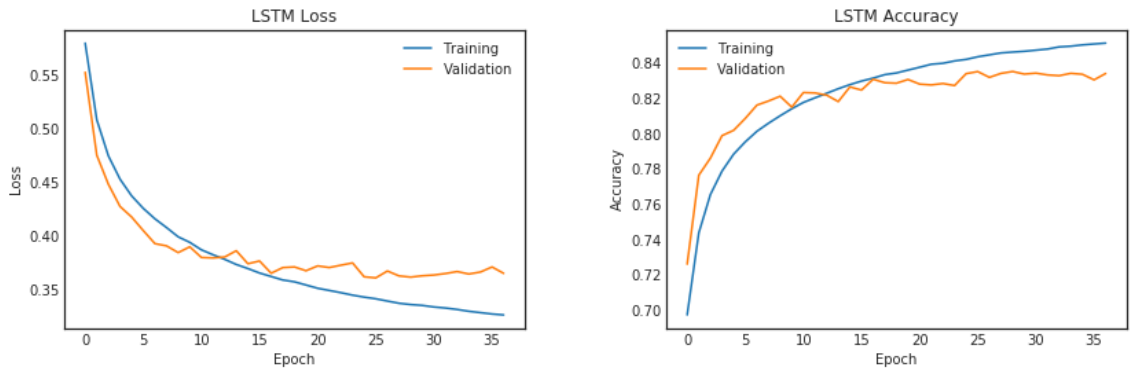
Figure 4: Comparison of ROC curves.

It is clearly visible that the performance of the benchmark model and the logistic regression is very similar. The LSTM model is the best performing classifier for false positive rates below a value of approximately 0.35. For higher false positive rates the XGBoost classifier has a slightly higher true positive rate.

Comparing all metrics and models it is clear that the LSTM network is in general the best performing model and it is therefore chosen to be the final estimator. The loss difference

between the benchmark model and the LSTM is $\Delta L_{log} = -0.3883$ and the difference in accuracy is $\Delta ACC = 0.14$.

In order to simulate the real world performance of the algorithm the test set is used once on the final estimator. The final estimator is trained again on a larger dataset because more data should increase the performance of the LSTM network. The training and validation sets are joined to create the new larger dataset. As done before with the LSTM, the joined dataset is split again into a training set and an epoch validation set that is used to validate the performance at every epoch. The development of the loss and the accuracy during the final training is shown in figure 5.



(a) (b)

Figure 5: Change in loss (a) and accuracy (b) of the LSTM network during training on the final joined dataset.

The development of the loss looks very similar to the previously shown change in figure 3(a) and the data starts to be overfit after a similar number of epochs. The final performance on the new epoch validation set is: $L_{loss} = 0.3650$, which is a promising result.

The model is now ready to be tested on the test set. The final LSTM network has a log loss (L_{log}) and an accuracy (ACC) of:

$$L_{log} = 0.3670$$
$$ACC = 0.83$$

on the test set. The loss is very close to the loss that was obtained earlier using the validation set and the accuracy has not changed. This is a strong indicator that the LSTM network will work efficiently on new and unseen data. Quora experimented themselves with deep learning techniques and received an accuracy on their on test between 0.86 and 0.87[23]. It is not described in detail on which dataset their models were trained and tested. Thus a direct comparison of the performance is impossible. However, these values can be used as a guideline. The accuracy of the classifier that was developed in this project is with, a value of 0.83, close to the performance of the LSTM networks tested by Quora.

It was initially planned to look at the performance of the classifiers used in the kaggle competition. However the competition unfortunately suffered from a data leak, which makes comparisons meaningless.

[23] `https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning`

5 Conclusion & Outlook

The goal of this project was to develop and train a classifier that can be used to find questions that are semantically equal. A benchmark was created based on the mentioned kaggle competition. An interesting observation is that simple models used on this text data, such as the benchmark model, already work rather well on their own. In this case the benchmark model receives an accuracy score of 0.69 on the validation dataset which is an improvement of 6% compared to a pure zero submission which would have resulted in an accuracy score of 0.63. During the development of this project two more advanced classifiers were trained that could possibly fulfill the project goal, a boosted decision tree classifier and a LSTM network. Some interesting predictions that were made on the validation dataset by these classifiers are shown in table 10.

Question 1	Question 2	Is duplicate	LSTM	XGBoost
Does ayurveda work??	Does Ayurveda actually work?	1	0.75790673	0.522238
Should people care more about doing the right thing, or doing things right?	Should people care more about doing the right thing, or doing things the right way?	1	0.950	0.198
How do I recover deleted messages in Facebook?	Can I recover my deleted Facebook messages?	1	0.6048215	0.929
How school teachers ought to help by?	I'm bad at math because of the way my high school teachers teach me. Is there any way I can get better at math without the help of teachers?	0	0.044	0.1999
What are the main problems in maglevs and Hyperloop?	What's known about the Hyperloop so far?	0	0.291	0.001
What do you think is the purpose of existence of human being in this vast universe?	What is the purpose of human existence in this vast universe?	0	0.99745917	0.928892

Table 10: Example predictions of the validation dataset. The last two columns show predicted probabilities that these questions are duplicates. The predictions of the LSTM network and the XGBoost Classifier are shown.

As is shown the LSTM and the XGBoost Classifier do usually predict similarity with different levels of confidence. Sometimes a classifier is also completely wrong as can be seen in row 2. An interesting result is shown in the last row. The data is not labelled as a duplicate but the questions seem to be identical. The dataset was labelled manually and this could just be a mistake. Interestingly both classifiers predict that this question is a duplicate with a high probability.

The LSTM network was chosen as the final estimator because it performed better in most cases. The LSTM accuracy and its loss are good enough to classify questions on an automated

basis. Classifiers like this could for example be used to add links between questions which have the highest similarity scores. However, the performance is not good enough to collect all unique topics on a single page. An accuracy of about 83% would introduce too many falsely classified questions that would be grouped together. Automated text classification at its current state of development can be seen as a step towards Quora's final goal but it can not fulfill all the requirements yet.

Several improvements could be implemented in the future to further boost the performance of these classifiers. First, more parameters could be tested. For example: two different numbers of LSTM units were tested. It would be feasible to test additional numbers that differ largely from the tested numbers.

In addition, as is explained in section 3.3, the performance does not change much with the change of the hyperparameters that were tested but it depends somewhat on luck. It will thus most likely make sense to train the model several times and use the average of the best models as the new prediction. The hyperparameters could be fixed during this time or they could be randomized in a certain range so that each model uses slightly different parameters.

Furthermore, even though the LSTM works better in general the XGBoost classifier has better predictions some of the time. This can for example be seen in table 10 in row 3 and 4. The LSTM is still able to predict the correct class but the XGBoost classification is more confident. Both classifiers seem to be able to make strong predictions for different types of question. This effect could be used to improve the prediction by stacking classification algorithms. The output of these two algorithms could for example be used as inputs to a third algorithm.

Better preprocessing and additional new features could also improve the performance of the XGBoost Classifier, which would be useful if the above mentioned stacking is used. An interesting candidate for an additional feature could for example be the tf-kld value [7].

References

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Corr*, abs/1301.3781, 2013. URL: http://arxiv.org/abs/1301.3781.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality.

[3] Tomas Mikolov, Scott Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: human language technologies (naacl-hlt-2013)*. Association for Computational Linguistics, May 2013. URL: https://www.microsoft.com/en-us/research/publication/linguistic-regularities-in-continuous-space-word-representations/.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735. eprint: http://dx.doi.org/10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: global vectors for word representation. In *Empirical methods in natural language processing (emnlp)*, 2014, pages 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[6] Steven Bird and Edward Loper and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[7] Yangfeng Ji and Jacob Eisenstein. Discriminative improvements to distributional sentence similarity. In *Emnlp*, 2013.