

Capstone Project

Machine Learning Engineer
Nanodegree

Ang Zhen Xuan

26th December 2017

Quora Question Pairs

I. Definition

Project Overview

Domain Background

Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly worded questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both groups in the long term.

Datasets and Inputs

Dataset obtained from: <https://www.kaggle.com/c/quora-question-pairs/data>

Input variables of the train.csv include:

Independent variables

- id - the id of a training set question pair
- qid1, qid2 - unique ids of each question (only available in train.csv): question1, question2

Dependent variables

- is_duplicate - the target variable, set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise.

The question1 and question2 text will be cleansed on (through stemming and stop words, etc) prior to the training of the machine learning models. Said model is trained on the training dataset to prevent implicit bias

infused into the trained model by 'peeking' into the test model¹. while testing is done on the testing dataset. The accuracy obtained from comparing the model procured answers and the labels on the testing set can be used as a decent gauge for the performance of the machine learning techniques adopted in training.

Presence of noise: Human labelling is also a 'noisy' and inherently subjective process, and reasonable people will disagree. As a result, the ground truth labels on this dataset should be taken to be 'informed' but not 100% accurate, and may include incorrect labelling.

Imbalanced classes: The imbalance brought out by labelled classes needs to be addressed as 149,263 examples are labelled with 1, and the majority examples are labelled with 0. This is an issue to be addressed either by oversampling/undersampling methods².

I originally planned to use the following pre-trained word embeddings (resources shared on Kaggle³):

1. Google's word2vec embedding from [this link](#)
2. Glove word vectors from [this link](#)
3. Facebook's fastText embeddings from [this link](#)

Only the word2vec embedding from Google was utilized since I was computing with limited resources (on dual-core CPUs).

Problem Statement

Currently, Quora uses a Random Forest model to identify duplicate questions. By tackling this natural language processing problem by applying advanced techniques to classify whether question pairs are duplicates, will make finding high quality answers to questions easier. This would result in an improved experience for Quora writers, seekers, and readers. I used the sklearn library's RandomForestClassifier to replicate a simple benchmark prediction model to be compared against.

The accuracy of the trained model must be able to quantify well enough (in the accuracy sense, to justify the use of the benchmarked Random

1 Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H. (2012). *Learning from data: a short course*. S.I.: AMLbook.com. Chapter 5 – Three Learning Principles.

2 Dealing with class imbalance while using CNNs.
http://www.academia.edu/8472416/Tackling_Class_Imbalance_with_Deep_Convolutional_Neural_Networks

3 Word embeddings resources shared on Kaggle -

<https://www.kaggle.com/c/quora-question-pairs/discussion/30286>

Forest classifier or any subsequent prediction model to be used after training) as well as be able to generalize well enough to new question pairs.

I adopted the Keras wrapper with Tensorflow as its backend to train the Manhattan distance LSTM⁴ (MaLSTM for short) model. The NLTK library was also used on the word vectors of the question pairs to help tokenize word into vectors, facilitating model training.

Metrics

It was mentioned that Quora uses a model based on Random Forests to detect similarities between questions. The competition requires participants to generate predicted probabilities for the submission file, I sought to minimise the error based on the weighted log-loss function instead of the conventional metric accuracy as we do not have a “untouched” test dataset⁵ with labels to train on.

I simulated a simple Random Forest model, and used it as a benchmark against the MaLSTM.

```
## accounting for the discrepancy of the class breakdown of training and testing data
def weighted_log_loss(y_true, y_pred, positive_train_labels, positive_valid_labels):
    a = positive_valid_labels/positive_train_labels
    b = (1-positive_valid_labels)/(1-positive_train_labels)
    score = []
    for pred, true in zip(y_pred, y_true):
        score.append(a*true*np.log(pred) + b*(1-true)*np.log(1-pred))
    return float(-np.sum(score)/len(score))
```

Figure 5: Weighted log-loss function used in training the RandomForest classifier

The weighted log-loss function was derived from the hold-out test set that I plan to use to evaluate the effectiveness of the prediction models. The test dataset has a label split of 83% non-duplicates and 17% of duplicates compared to the 63-37 split in the training dataset. Accounting for the difference in distributions between both datasets, the weighted log-loss function was derived based on the Bayes’ Theorem⁶.

4 Original paper published for the MaLSTM model - http://www.mit.edu/~jonasm/info/MuellerThyagarajan_AAAI16.pdf

5 Data snooping & related issues - <https://work.caltech.edu/library/173.html>

6 Accounting for differences in training and test dataset’s distributions - <https://swarbrickjones.wordpress.com/2017/03/28/cross-entropy-and-training-test-class-imbalance/>

```
def weighted_log_loss(y_true, y_pred):
    a = 0.165/0.37
    b = (1-0.165)/(1-0.37)
    score = a*y_true*K.log(y_pred+0.00001) + b*(1.0 - y_true)*K.log(1.0 - y_pred+0.00001)
    return -K.mean(score)

def exponent_neg_manhattan_distance(left, right):
    return K.exp(-K.sum(K.abs(left-right), axis=1, keepdims=True))
```

Figure 6: Weighted log-loss function re-written for the Keras' library - used in training the Siamese maLSTM classifier

$$\exp(-\|h^{(left)} - h^{(right)}\|_1)$$

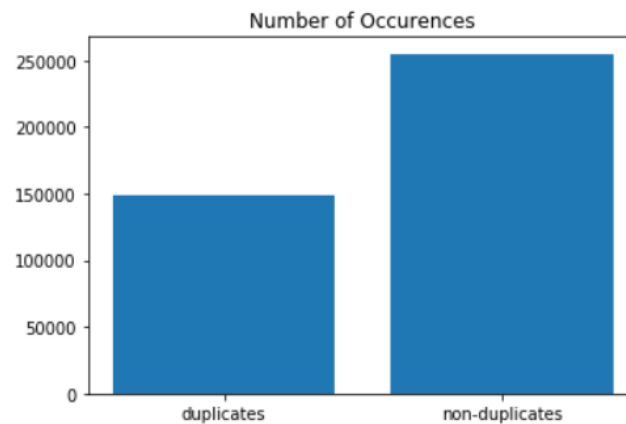
Figure 7: maLSTM similarity function

The similarity function is used in the MaLSTM model to capture the similarity between questions in terms of their semantic and syntatic behaviour. Reducing the total weighted log loss is still the main aim of the MaLSTM model.

II. Analysis

Data Exploration & Exploratory Visualization

Referencing Figure 2, the percentage of duplicates to non-duplicates is as shown:



Number of duplicates in the training dataset: 149263
 Number of non-duplicates in the training dataset: 255027
 Percentage of duplicates in training dataset: 36.92%
 Percentage of non-duplicates in training dataset: 63.08%

Figure 1: Amount and Percentage of duplicates to non-duplicates in the training dataset

This puts me in good stead with regards to the distribution of the training dataset. Next, I looked into the null values contained within the training

distribution. This was easily achieved using **info()** method on my **train** DataFrame.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 404290 entries, 0 to 404289
Data columns (total 6 columns):
id                404290 non-null int64
qid1              404290 non-null int64
qid2              404290 non-null int64
question1         404289 non-null object
question2         404288 non-null object
is_duplicate      404290 non-null int64
dtypes: int64(4), object(2)
memory usage: 18.5+ MB
```

Figure 2: Checking the null values that may need to be handled

As shown in the figure above, both the question1 and question2 rows show a count of non-null objects that is less than the total number of question pairs in the training dataset (404289 and 404288 respectively). This proves that there are two null values in the question2 column while there exists one null value in the column of question1.

```
train[train.isnull().any(axis=1)]
```

	id	qid1	qid2	question1	question2	is_duplicate
105780	105780	174363	174364	How can I develop android app?	NaN	0
201841	201841	303951	174364	How can I create an Android app?	NaN	0
363362	363362	493340	493341	NaN My Chinese name is Haichao Yu. What English na...		0

```
test[test.isnull().any(axis=1)]
```

	test_id	question1	question2
379205	379205	How I can learn android app development?	NaN
817520	817520	How real can learn android app development?	NaN
943911	943911	How app development?	NaN
1046690	1046690	NaN How I what can learn android app development?	
1270024	1270024	How I can learn app development?	NaN
1461432	1461432	NaN How distinct can learn android app development?	

Figure 3: Making sure the non-null rows are actually NaN values

Algorithms & Techniques

Neural networks are adaptable, and have been gaining traction in the field for handling word vectors yielding comparable results to classical measures like Cascading Features and Shallow Joint Training for Natural Language Processing problems⁷.

My original intention of training a model based on Convolutional Neural Networks (CNN) over RNNs was due to the fact that RNNs are slower and ficker to train, and sequencing may not be that important⁸ as 'feelings' detection (such as happiness or sadness) in corpora may be more essential.

Method	r	ρ	MSE
Illinois-LH (Lai and Hockenmaier 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al. 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al. 2014)	0.8268	0.7721	0.3224
ECNU (Zhao, Zhu, and Lan 2014)	0.8414	–	–
Skip-thought+COCO (Kiros et al. 2015)	0.8655	0.7995	0.2561
Dependency Tree-LSTM (Tai, Socher, and Manning 2015)	0.8676	0.8083	0.2532
ConvNet (He, Gimpel, and Lin 2015)	0.8686	0.8047	0.2606
MaLSTM	0.8822	0.8345	0.2286

Figure 8: Referencing the MSE column, it has the lowest error rate compared to the other models based on the SICK semantic similarity task.

However, after further research, I realised that the Siamese Manhattan LSTM (MaLSTM)⁹ provides a rather straightforward approach to the problem of sentence similarity. With reference to Figure 1, the MaLSTM has outperformed its counterparts (similar semantic handling models) by obtaining the lowest mean squared error on the SICK (Sentences Involving

7 Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing. *Proceedings of the 25th international conference on Machine learning - ICML 08*. doi:10.1145/1390156.1390177

8 Reasons relating to picking CNN over RNN -

<https://datascience.stackexchange.com/questions/11619/rnn-vs-cnn-at-a-high-level>

9 How to predict Quora Question Pairs using Siamese Manhattan LSTM -

<https://medium.com/mlreview/implementing-malstm-on-kaggles-quora-question-pairs-competition-8b31b0b16a07>

Compositional Knowledge) dataset – usually used to gauge performances of models with regards to accounting for syntactic and semantic issues¹⁰. It has also proven to perform well especially in tasks such as semantic similarity, which has been essential to solving Natural Language Processing problems in recent times.

Benchmark

I seek to minimise the weighted log loss that was defined in the Metrics section, as well as minimizing the error achieved on the public and private leaderboards as a gauge of my attempt relative to the Kaggle leaders.

III. Methodology

Data Preprocessing

I replace the null values with empty strings to ensure that the null values are handled in a proper manner – without compromising of potential predictive power in the case of empty inputs. I also ensure that the two question rows tally with the amount of null values.

```
train = train.fillna("")
test = test.fillna("")

# Verify that rows with null values have been removed
train.info()
test.info(null_counts=True)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 404290 entries, 0 to 404289
Data columns (total 6 columns):
id                404290 non-null int64
qid1              404290 non-null int64
qid2              404290 non-null int64
question1         404290 non-null object
question2         404290 non-null object
is_duplicate      404290 non-null int64
dtypes: int64(4), object(2)
memory usage: 18.5+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2345796 entries, 0 to 2345795
Data columns (total 3 columns):
test_id           2345796 non-null int64
question1         2345796 non-null object
question2         2345796 non-null object
dtypes: int64(1), object(2)
memory usage: 53.7+ MB
```

Figure 4: Replacing N.As with empty strings

10 The SICK dataset - <http://cltc.cimec.unitn.it/composes/sick.html>

Implementation

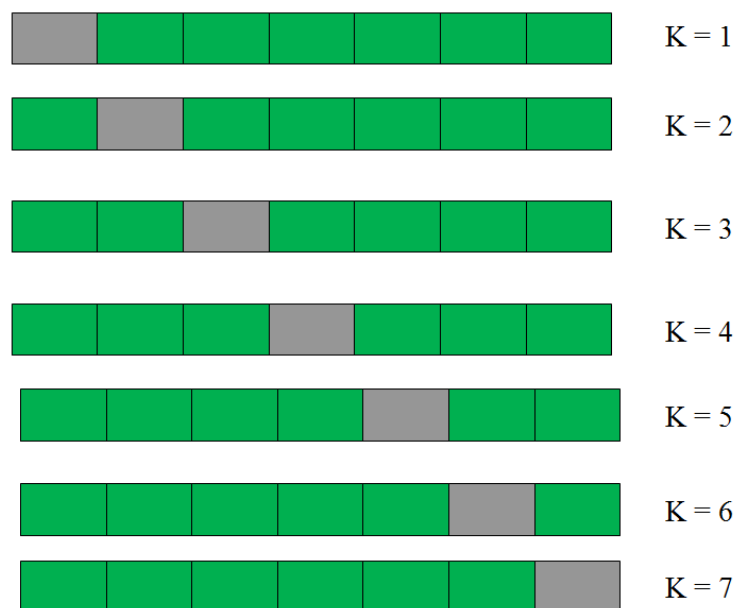
I started building the benchmark Random Forests classifier by data engineering on a couple of features.

Features include:

1. The number of words in a question
2. The average word length in a question
3. The total number of letters in a question
4. The number of sentences that start off with a capitalized letter in a question
5. The Jaccard index/Similarity Coefficient - the Bag-Of-Words variant¹¹ between two questions
6. Levenshtein distance between two questions¹²

I used the RandomForestClassifier¹³ module to train my prediction model, with only the number of Random Forests (n_estimators) as a tunable hyperparameter.

I adopted the grid search process on a 10-fold cross validation dataset. The scikit-learn library GridSearchCV makes it easy to tune the hyperparameters while validating the results against the split sets.



11 Explanation of the difference between the Bag-of-Words and Frequency Distribution variants of the Jaccard index - <https://nickgrattandatascience.wordpress.com/2017/12/31/bag-of-words-and-frequency-distributions-in-c/>

12 Levenshtein distance as covered by Wikipedia - https://en.wikipedia.org/wiki/Levenshtein_distance

13 RandomForestClassifier - <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

Figure 9: k-fold (k=7) cross validation training - the grey areas are the validation sets while the green blocks sum up to be the training dataset in each iteration

Refinement

Looking at the rather disappointing results of my Random Forest classifier, and the trouble in which the Keras library have eased to train neural nets, I decided to pivot to a deep learning Siamese maLSTM classifier built by word embeddings as features. In addition, the classical features do not provide explainable insights to the results that I have obtained after training

I adopted Google's corpus as their pre-trained word embeddings have been trained on three million english words which should constitute as extensive for my use case. The pre-trained embeddings also mean that I do not have to train my own word embedding which may take an awful of long time for a model powered by a CPU.

Firstly, I had to define the architecture of the Siamese maLSTM model, which was discussed extensively in this Medium article¹⁴.

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	(None, 213)	0	
input_4 (InputLayer)	(None, 213)	0	
embedding_2 (Embedding)	(None, 213, 300)	36427800	input_3[0][0] input_4[0][0]
lstm_2 (LSTM)	(None, 50)	70200	embedding_2[0][0] embedding_2[1][0]
merge_2 (Merge)	(None, 1)	0	lstm_2[0][0] lstm_2[1][0]
Total params: 36,498,000 Trainable params: 70,200 Non-trainable params: 36,427,800			

Figure 9: Architecture of the Siamese LSTM model

I proceeded to process the inputs for the model - generating the word matrix for my word embeddings. This is done in a couple of steps:

1. Splitting the training dataset (**train.csv**) into a 80-20 split, using the 20% as a validation set of sorts to ensure better generalization can be achieved to achieve better results on the holdout test set on Kaggle.

¹⁴ Architecture discussed in this article -

<https://medium.com/mlreview/implementing-malstm-on-kaggles-quora-question-pairs-competition-8b31b0b16a07>

2. Cleaning the datasets through importing NLTK's stopwords as well as a couple of regex formatting to ensure words are consistent (hasn't is converted into has not for example).
3. Iterate through every sentence in both columns of both the training and testing datasets.
4. Since word vectors cannot be inputted and trained upon, they have to be converted into a numerical representation before training
5. Keep a dictionary record of every word in that has appeared, as well as holding another array with the numerical representations of each and every word of my dictionary.
6. Filling the matrix with each word's 300 dimensional representation. The matrix should be a $|D|^{15} \times |300|$

IV. Results

Model Evaluation & Validation

The final model's parameters were as such:

- 50 hidden layer nodes
- Batch size of 128
- 10 epochs
- Adam optimizer, clipping the norm at a value of 0.9

Model	Metric	Weighted Log-Loss Error	Kaggle Leaderboard (LB) score	
			Public	Private
Random Forests Classifier w/ 10-fold CV		2.81428114447	0.59123	0.58364
Siamese MaLSTM w/ pre-trained word embeddings		0.290278734785	0.34697	0.35206

Figure 10: Comparing metrics & results

¹⁵ It should equate to the number of words collected by the dictionary in step 2.

Justification

The results were compared for both models with reference to Figure 10. A marked improvement in every metric that I have sought to compare for my original benchmark model and the refined model was observed.

I believe with the huge reduction in the weighted log-loss error which accounts for the imbalance in the dataset, as well as the huge drop in the LB score on Kaggle, I think the pivoting of model from a Random Forests classifier to a Siamese MaLSTM that better captures the similarity or lack thereof between questions was a justified decision. Last but not least, I managed to achieve a top half position with that LB score.

v. Conclusion

Reflections

Natural Language problems have been and are still a challenging form of issue to deal with. Even though much progress have been established within the field of Deep Learning and Machine Learning, great results that we may have achieved - may not mean that our prediction models have captured the semantic, syntactic meaning behind words.

Improvements

In conclusion, after rigorous testing and tuning of the prediction model's parameters, I believe there are a few points we can conclude for the end model:

1. Firstly, transfer learning could be applied to our initial weights - pre-train MaLSTM on separate sentence-pair data provided in the SemEval 2013 Semantic Textual Similarity paper instead of randomly drawing weights from a Gaussian distribution. This has **proven to be a superior starting point compared to random initialization**¹⁶.
2. Attempted dropout to **prevent overfitting**, but I am not too sure of its application within my defined LSTM layers.
3. **More data usually provides more robust and reliable prediction models.**

¹⁶ SemEval 2013 Semantic Textual Similarity - <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=36B3188663E26B1D311592D8757A11B7?doi=10.1.1.310.7053&rep=rep1&type=pdf>

- a. Generate possible new data, be it manually or through generative deep learning systems such as the GAN (Generative Adversarial Networks) to be used in the training processes.
 - b. Possibly replace words with their synonyms to obtain more training data.
- 4. The word embeddings from Google that I have utilized may not have been the best contextual corpus to derive weights from as Google's corpus was based on news articles and reports. This corpus may not be providing totally relevant context to my corpus of sentences from a diversity of categories on Quora. **This is likely to lead to a better accuracy with more relevance to the broadness of categories Quora is likely to cover.**
- 5. Lastly, the classical features could have been expanded on as the number of features were limited - might not have resulted in a representative Random Forests classifier.

All in all, the points discussed above may or may not lead to a better score or accuracy, but they do warrant as potential points of improvements that I could look into.