Udacity Machine Learning Nanodegree Capstone

Quora Duplicate Question Detection

Raahul Seshadri

April, 2017

Contents

1 Definition

1.1 Project Overview

Websites like Quora[1], StackExchange[2] network (includes the popular programming website StackOverflow[3]) allow the users to post questions, and the entire community can answer those questions. Ideally, every unique question should be present just once in the system, so that every answer to those questions are present at once place.

However, users are prone to posting duplicate questions, because not all of them check if the question they're asking has already been asked by someone else. This necessitates having an automated duplicate question checker that can check if a question is a duplicate. Whenever the user tries to post a new question, the system can suggest an existing one for perusal.

This project was inspired by Quora's Kaggle challenge[4]. A dataset of question pairs, manually tagged as duplicate or not by human reviewers, has been provided[5] by Quora to train on.

For the purpose of this project, the inputs are strictly assumed to be in English.

1.2 Problem Statement

The problem can be stated as follows:

Given a pair of question texts, detect if they are duplicate or not.

For example, given the following two questions:

Q1: What is the average salary in India?
Q2: What is the average salary in the United States?

The system could flag them as either duplicates or not-duplicates. The steps to achieving our duplicate detector are as follows:

1. Download and pre-process the Quora training dataset
2. Split the Quora training dataset into 90% training and 10% test set.
3. In the 90% training set, further 10% will be used as a validation set.
4. Extract usable features from the dataset.
5. Train a binary classifier to differentiate between duplicates/non-duplicates.
6. Provide a command-line interface so that users can check for duplicates from Quora's test set, or provide his own questions.

[1] https://www.quora.com
[2] https://stackexchange.com
[3] https://stackoverflow.com
[4] https://www.kaggle.com/c/quora-question-pairs
[5] https://www.kaggle.com/c/quora-question-pairs

1.3 Metrics

Being a binary classification problem, the metric is going to be accuracy:

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{total samples}}$$

This accuracy will be measured on the test set (10% of the total dataset).

1. **False negatives**, where a duplicate pair is not detected as such, will lead to degraded user experiences. However, it is also rather harmless, since no information is lost. There is still an opportunity to manually flag them as duplicates, or build such a feature on the platform that is using the duplicate detector.

2. **False positives**, where a question pair that's not a duplicate, but is marked as such, is more dangerous. It can lead to very degraded user experience, since it directly hinders the user trying to perform an action. It is better for the system to err on the side of false negatives than false positives.

2 Analysis

2.1 Data Exploration

Below is the head (first 20 entries) of the Quora dataset:

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0
5	5	11	12	Astrology: I am a Capricorn Sun Cap moon and c...	I'm a triple Capricorn (Sun, Moon and ascendan...	1
6	6	13	14	Should I buy tiago?	What keeps childern active and far from phone ...	0
7	7	15	16	How can I be a good geologist?	What should I do to be a great geologist?	1
8	8	17	18	When do you use シ instead of し?	When do you use "&" instead of "and"?	0
9	9	19	20	Motorola (company): Can I hack my Charter Moto...	How do I hack Motorola DCX3400 for free internet?	0
10	10	21	22	Method to find separation of slits using fresn...	What are some of the things technicians can te...	0
11	11	23	24	How do I read and find my YouTube comments?	How can I see all my Youtube comments?	1
12	12	25	26	What can make Physics easy to learn?	How can you make physics easy to learn?	1
13	13	27	28	What was your first sexual experience like?	What was your first sexual experience?	1

The columns correspond to the following features:

1. **id:** ID to identify the question pair uniquely
2. **qid1:** ID of the first question in the question pair
3. **qid2:** ID of the second question in the question pair
4. **question1:** Text of the first question in the question pair
5. **question2:** Text of the second question in the question pair
6. **is_duplicate:** Quora reviewer's decision on whether the question pair is duplicate (1) or not (0)

2.1.1 Why do we have question IDs for each pair?

There are two columns of interest, "qid1" and "qid2" that are of interest. The algorithm that we're going to design will only tell us if a given pair of question is a duplicate. However, given a question, we'd also like to find out all the duplicates of it in the system.

For example, if question "1" is a duplicate of question "2", and question "2" is a duplicate of "3", then a system should also know that "1" and "3" are duplicates. Data structures like "union find" allow us to do that. However, this is not in scope of the algorithm that we'll design, but the responsibility of the higher system that will use this duplicate detector.

Which is why the question IDs won't be used as a feature for the duplicate detector, but is still important for the system using the duplicate detector.

2.1.2 Summary statistics

Below are some basic summary statistics:

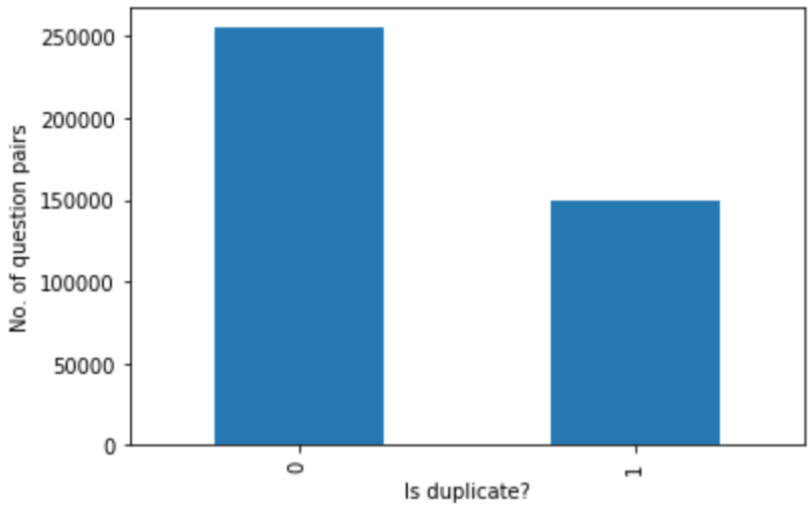
1. **Total entries:** 4,04,290
2. **Total positive entries:** 1,49,263
3. **Total negative entries:** 2,55,027
4. **Percent positive entries:** 36.91%
5. **Percent negative entries:** 63.08%

We have a sample with more negative than positive examples.

2.2 Exploratory Visualization

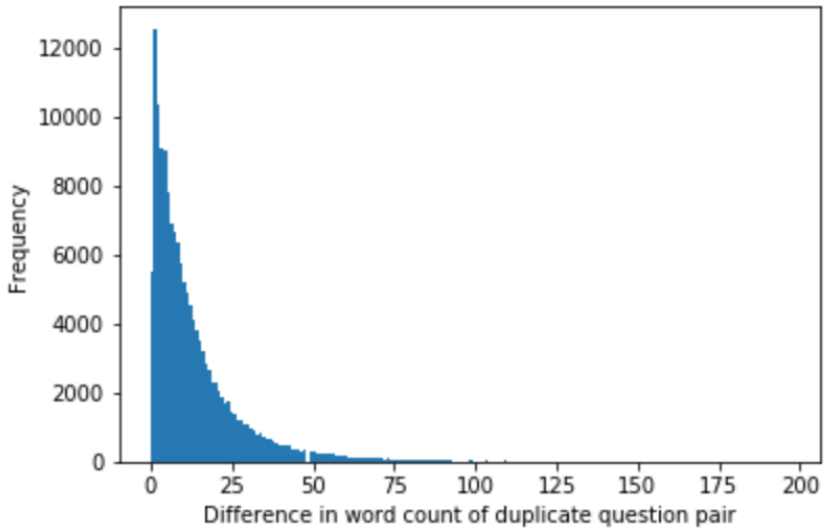
2.2.1 Summary visualization

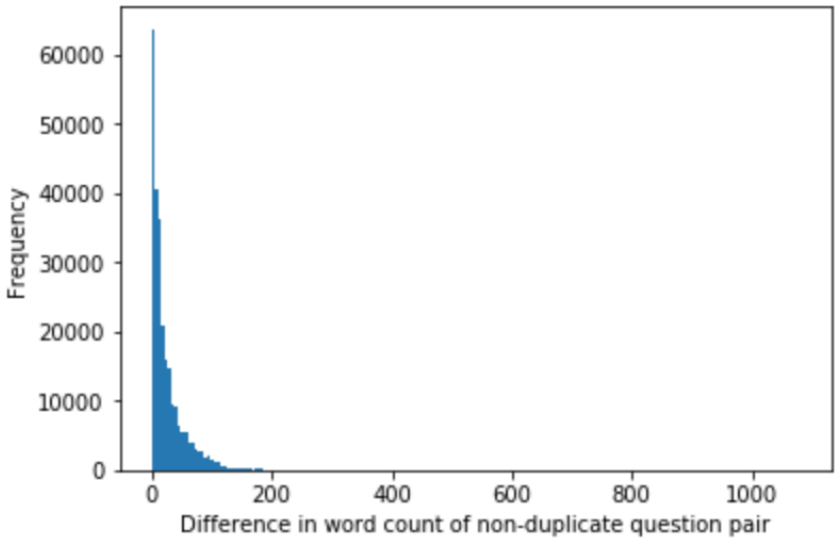
Let's look at the class distribution (duplicate or not). This is the same from our summary statistics.



2.2.2 Word count differences

Let's look at if word count has any bearing on whether a question is marked as duplicate or not, or can we discern any important facts/patterns.





1. For duplicates, the word count difference tapers smoothly
2. For non-duplicates, there is a sharp decrease in the 2nd bin
3. There are question pairs that have word count difference 75 and above, and are still duplicates. So simple bag of words model will not work. *This is the most important conclusion.*

2.2.3 Jaccard index

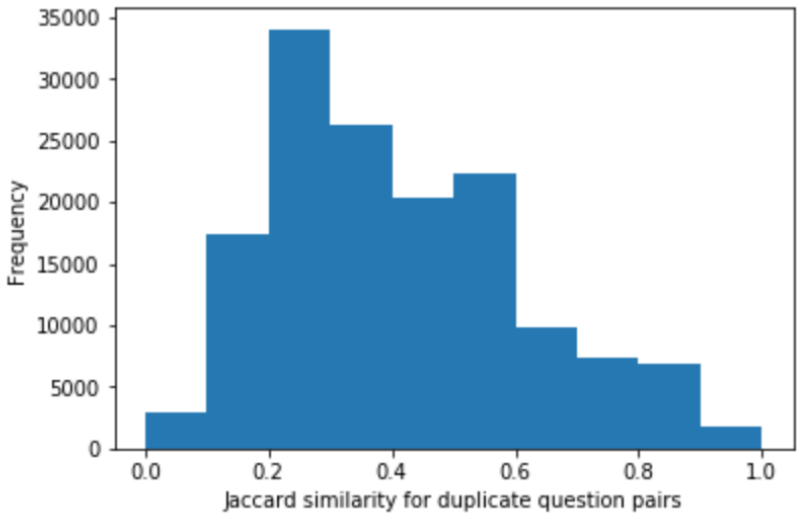
To explore a baseline model, let's look at the bag-of-words similarity of duplicate and non-duplicate question pairs using the Jaccard Index[6]. The relevant formula is:

$$J(A, B) = \frac{A \cap B}{A \cup B}$$

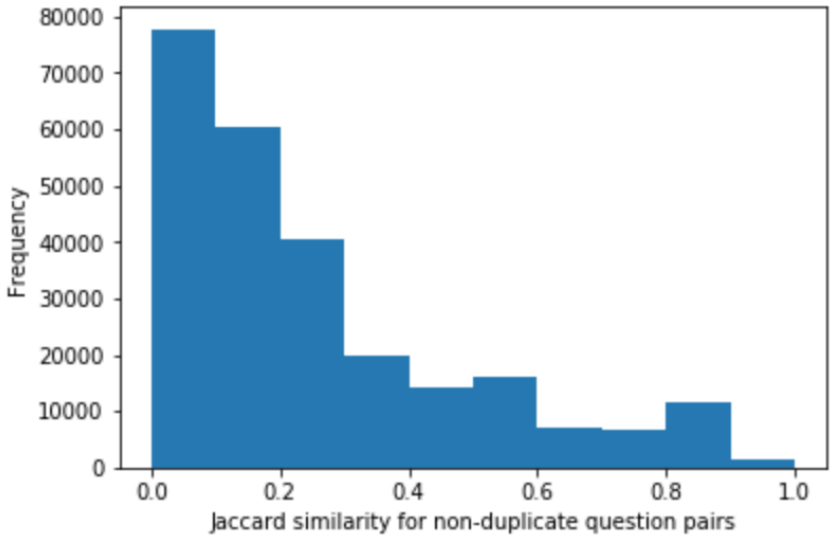
In our case, A and B refer to individual words in the question pairs. So, the formula basically is:

$$J(A, B) = \frac{\text{No. of common words in question1 and question2}}{\text{No. of unique words in question1 and question2 combined}}$$

Let's look at the Jaccard similarity of duplicate and non-duplicate question pairs separately.



[6]https://en.wikipedia.org/wiki/Jaccard_index



1. For duplicates, Jaccard similarity is like a gamma distribution. Jaccard is bad at predicting a pair is a duplicate.

2. For non-duplicates, the distribution is again gamma but with a right skew. So Jaccard positively gives low scores for most of the non-duplicates.

We had previously concluded from the word counts that bag-of-words similarity will not be a good method of differentiating between duplicates and non-duplicates, and the two graphs above confirm the fact.

2.3 Algorithms & Techniques

2.3.1 Feature extraction

The inputs to our algorithm are two variable length strings. In NLP (Natural Language Processing), there are two primary ways of extracting features from text, after the input has been word-tokenized, that is being split into individual words:

Bag of words:
Here, every word is assigned a unique number. The input to any ML algorithm then becomes a one-hot encoded array of numbers that correspond to these words.

For example, if the input corpus has two words: “hello” and “world”. And we assign 0 to “hello” and 1 to “world”. Then, the word “hello” would be represented as [1 0] while “world” would be represented as [0 1]. (one hot encoded).

The number of features become equal to the number of unique words in the corpus (training set).

Word vectors:
The problem with one hot encoding is that we come up with very large feature vectors, since they're one hot encoded. This does not scale to large corpora.

Instead of a sparse array, word vectors convert input words into a fixed-length vector. A common vector size is 300. Thus, every word in the input corpus gets converted into a dense feature vector of length, say, 300, regardless of how many words there are in the corpus.

As opposed to bag of words, word vectors have to be trained on a corpus, before they can be used to extract features from different corpora. Thankfully, there are a lot of pre-trained models for word vectors available.

Two popular algorithms to generate them are Word2Vec[7] and GloVe[8]. spaCy[9] comes with a GloVe model[10] of 300-dimensional vectors trained on the Common Crawl corpus. This is the Python library that we will use to do all our processing.

Naturally, I'll be using the word vectors approach, since it also allows me to train for words that don't appear in the Quora training corpus, but a similar word does.

2.3.2 Pooling

Given a pair of questions, the questions themselves can be of different sizes. If we get word vectors for each word in the question, then we get a matrix of size $m \times 300$, where we're assuming that 300 is the word vector size and m is the number of words in the question.

However, for a pair, m could be different. Thus, we need a way to normalize this. There are two ways[11]:

1. **Max pooling:** Take the maximum of each column in the $m \times 300$ matrix to get a 1×300 matrix.

2. **Mean pooling:** Take the mean of each column in the $m \times 300$ matrix to get a 1×300 matrix.

Max pooling is reported to perform better[12]. However, I've decided to concatenate mean and max pooling to get a matrix 1×600 in dimension. Thus, every question is converted into 600-dimensional vector.

[7] https://en.wikipedia.org/wiki/Word2vec
[8] https://nlp.stanford.edu/projects/glove/
[9] https://spacy.io
[10] https://spacy.io/docs/usage/word-vectors-similarities
[11] https://explosion.ai/blog/quora-deep-text-pair-classification#example-neural-network
[12] https://explosion.ai/blog/quora-deep-text-pair-classification#results

2.3.3 Neural Networks

We will be using Neural Networks to create our model. Neural Networks, especially deep ones, have lately become popular in the field of NLP.[13] [14] [15]

A neural network learns to model functions, just like supervised algorithms. However, the deeper the neural network (deeper meaning more number of layers), the more complex characteristics it can detect in the input.

A downside to neural networks is that it is incredibly difficult to correctly design them. The network structure itself has many possibilities, and then we also have a number of hyperparameters like epochs, depending on the optimizer being used. They also take a long time to train.

2.3.4 Improving Neural Network Accuracy

There are several techniques in neural networks to improve the accuracy of the network, or to reduce overfitting.

Dropout:
Here, random units inside of the network are switched off in every epoch. Thus, the network is forced to learn redundant representations of the input. This, in totality, avoids overfitting. Dropout only happens during training, and is switched off when running the model in production.[16] [17]

Due to the low amount of training data that we have, I'm not using dropouts in my network.

Batch Normalization:
A Batch Normalization layer shifts the inputs from the previous layer to have zero-mean and unit-variance. This prevents data flowing in the network to not become too big or too small. It is said to result in higher accuracy and faster learning (convergence).[18] To try this out, I trained variants of my network with Batch Normalization on and off.

2.4 Benchmark

In the field of Information Retrieval, there is a problem called Near Duplicate Detection.[19] Jaccard Similarity of bag-of-words ($k = 1$ shinglings) is one way of finding near duplicates.[20]

[13] http://colah.github.io/posts/2014-07-NLP-RNNs-Representations
[14] https://www.quora.com/How-are-neural-networks-used-in-Natural-Language-Processing
[15] https://nlp.stanford.edu/projects/DeepLearningInNaturalLanguageProcessing.shtml
[16] https://www.quora.com/How-does-the-dropout-method-work-in-deep-learning
[17] https://www.cs.toronto.edu/ hinton/absps/JMLRdropout.pdf
[18] https://www.quora.com/Why-does-batch-normalization-help
[19] https://nlp.stanford.edu/IR-book/html/htmledition/near-duplicates-and-shingling-1.html
[20] http://stackoverflow.com/questions/23053688/how-to-detect-duplicates-among-text-documents-and-return-the-duplicates-similar

Since we split Quora dataset into 10% test set, this benchmark model would be run on the test set. This baseline model gives an accuracy of 65.153%.

3 Methodology

3.1 Data Pre-processing

The Quora training dataset *train.csv* is assumed to be in the same directory as the Jupyter Notebook.[21] The structure of the dataset is as in the exploration.

3.1.1 Feature selection

Out of all the columns in the dataset, we will use the following features:

1. **question1:** Text of the first question in the pair
2. **question2:** Text of the second question in the pair

And the target variable:

1. **is_duplicate:** Whether the question pair is a duplicate or not. 1 if it is, 0 if it is not.

3.1.2 Feature transformation

A question from a question pair would go through the following transformations. For the examples below, we'll assume that the input text is "This is a sentence'.'

Step 1: Tokenize text into words
Using spaCy, we will convert a text like "This is a sentence" into words (tokens) like: "This" "is" "a" "sentence"

Step 2: Get GloVe vectors for each word
For every word, we get a GloVe vector of length 300. Thus, we get a matrix of size *no. of tokens* $\times 300$. In the case of our example sentence, it will be a matrix of size 4×300. This looks as shown below:

This	0	0	...	0
is	0	0	...	0
a	0	0	...	0
sentence	0	0	...	0

Step 3: Mean pooling
The $n \times 300$ matrix is now converted into a 1×300 matrix by taking the mean

[21] Quora De-duplication.ipynb

of all the columns.

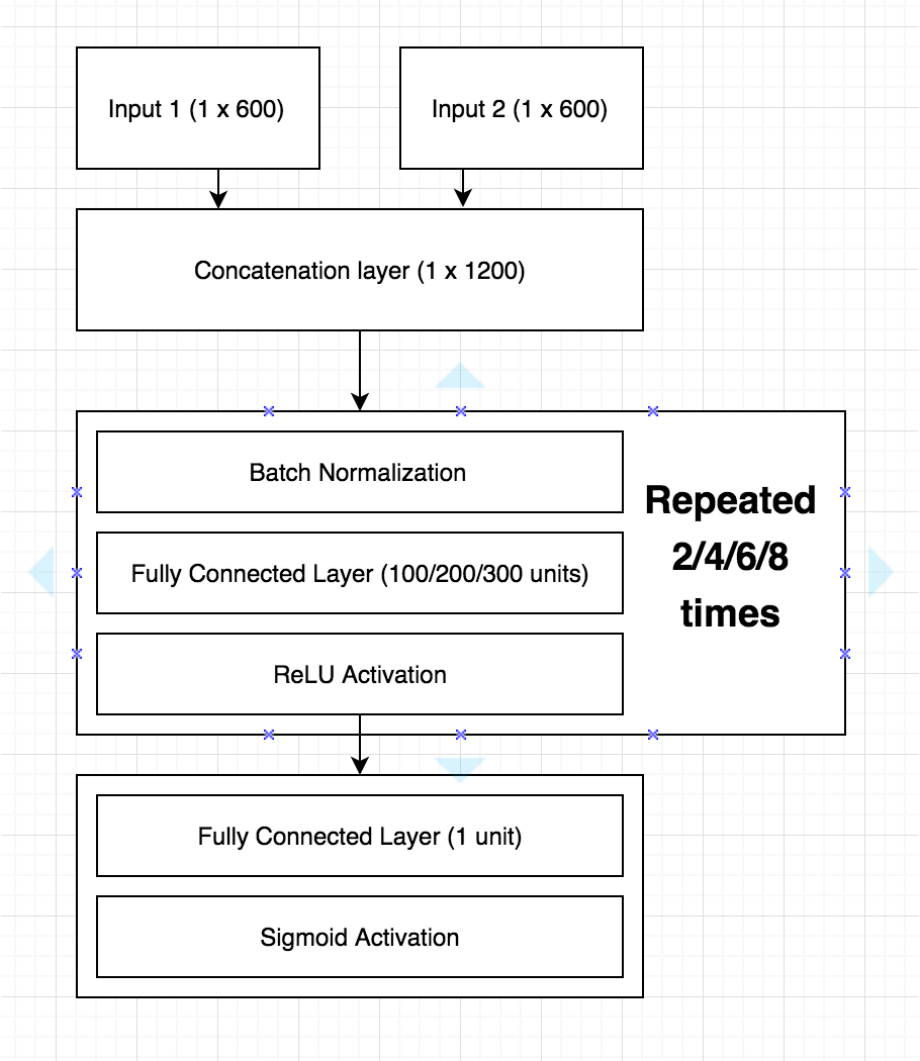
Step 4: Max pooling
The same $n \times 300$ matrix is converted again into a 1×300 matrix, but by taking the max from all the columns.

Step 5: Concatenate mean and max. pooling
The mean and max pool are concatenated into a matrix of size 1×600.

We repeat the above step for every question in every question pairs. This concludes our data pre-processing step. For our neural network, we will have two vectors of size 600 as inputs.

3.2 Implementation & Refinement

3.2.1 Structure of the Neural Network



The following are the characteristics of this network:

1. There are two inputs to the network, which are the 1×600 feature for each of the question in a pair.
2. These inputs are concatenated into one layer of size 1×1200.
3. A learning unit consists of a dense layer with ReLU activation, preceded by batch normalization.

4. The final layer is a dense layer with 1 neuron and sigmoid activation. This reflects the probability that a question pair is a duplicate.

To find the right network topology that will improve accuracy, we will try a couple of combinations.

1. Number of training units: 2, 4, 6, 8
2. Number of neurons in the dense layer: 100, 200, 300
3. Batch normalization: True, False

This gives rise to 24 combinations. We will train the neural network on all of those topologies. The number of epochs is set to 25, because that is the maximum I have the resources to train 24 neural networks on. However, the training and validation accuracy of each epoch is put in a log, so that we can always check later for patterns of overfitting.

3.2.2 Loss & Optimization

Binary crossentropy is used as a loss metric, since it's a binary classification problem. I used the adam optimizer, so as to avoid having to tune a lot of hyperparameters related to the optimizer by hand.

3.2.3 Code Implementation

To implement this neural network, we will use Keras[22], which is a library that sits on top of Theano[23] and TensorFlow[24]. We will be using the Theano backend.

A neural network is created for all 24 combinations. A single function automatically generates a neural network given a combination. The naming convention for each model is "*no_neurons__batch_norm__no_units*". Here's an example:

1. **No. of training units:** 2
2. **No. of neurons in dense layer:** 200
3. **Batch normalization:** True

In this case, the model will be named "200_True_2". The model will be saved to disk as "200_True_2.h5", and the log of training would be saved as "200_True_2.h5.log".

[22] https://keras.io/
[23] http://deeplearning.net/software/theano/
[24] https://tensorflow.org

3.2.4 Final performance

The model that performed the best, with test accuracy of **82.783%** has the following characteristics:

1. Number of training units: 6
2. Number of neurons in the dense layer: 300
3. Batch normalization: True

The baseline solution had an accuracy of 65.153% on the test set.

3.2.5 Performance of intermediate models

Let's look at how the 23 other neural network combinations performed.

Batch normalization: OFF

ROWS: No. of layers
COLS: No. of neurons in each layer

	100	200	300
2	0.7838	0.7866	0.7847
4	0.7834	0.7808	0.7876
6	0.7853	0.7782	0.7818
8	0.7860	0.7758	0.7738

Batch normalization: ON

ROWS: No. of layers
COLS: No. of neurons in each layer

	100	**200**	**300**
2	0.8130	0.8220	0.8231
4	0.8167	0.8221	0.8261
6	0.8174	0.8216	0.8278
8	0.8205	0.8244	0.8245

The following observations can be made:

1. The best performance with batch normalization OFF is **78.76%**. With ON, it's **82.78%**. There's a **4.02%** improvement.
2. With batch normalization ON, the lowest test accuracy is **81.30%**, and the highest test accuracy is **82.78%**. Tuning of hyperparameters has lead to a **1.48%** improvement in accuracy.

4 Results

4.1 Model Evaluation, Validation, and Justification

As mentioned previously, the model that performed best had the following characteristics, and a final test accuracy of **82.783%**:

1. Number of training units: 6
2. Number of neurons in the dense layer: 300
3. Batch normalization: True

The baseline solution had an accuracy of 65.153% on the test set. Thus the neural network solution gave an improvement of **17.63%** over the benchmark (that just naively compared bag of words and calculated the Jaccard index).

This final model had the following characteristics after 25 epochs (this data is obtained from "300_True_6.h5.log"):

1. Training accuracy: 91.300%
2. Validation accuracy: 82.370%
3. Test accuracy: 82.783%

The training accuracy is higher than the validation/test accuracy, so the model is slightly over-fitting. The validation and test accuracy are close to each other, so the model does generalize well to unseen data. Given the accuracy score, we can say that the model has helped in solve the problem.

With more amount of training data, the accuracy can be improved further. With significantly more data, custom GloVe vectors can be trained instead of using a pre-trained model. This would ensure that the vectors reflect the nature of the data the duplicate detector will run on.

5 Conclusion

5.1 Free-form Visualization

The most important quality of interest is the ability to recognize entities in the question. I've come up with these questions myself as an experiment. For

example, Quora reviewers consider the following to be NOT duplicate:

Question 1	What is the salary in the US?
Question 2	What is the salary in Germany?
Human opinion	NOT DUPLICATE
Model's prediction	NOT DUPLICATE

The reason they're not duplicates is because they talk about two different countries, even though most of the keywords in the question are the same. Our model gets it right.

On the flipside, let's look at a pair of question that ask the same thing but in different words:

Question 1	How to learn programming?
Question 2	How do I become good at programming?
Human opinion	DUPLICATE
Model's prediction	DUPLICATE

Question 1	Where do I go in Germany?
Question 2	What are some good places to visit in Germany?
Human opinion	DUPLICATE
Model's prediction	DUPLICATE

Word vectors helps our model recognize the basic intents behind each question, and thus categorize similar words as one. Let's change the name of the country in the above example, and the question is no longer classified as duplicate. Thus the names of entities do matter a lot to the model.

Question 1	Where do I go in Germany?
Question 2	What are some good places to visit in India?
Human opinion	NOT DUPLICATE
Model's prediction	NOT DUPLICATE

One problem of using pre-trained word vectors, is that they can flag entities as similar if they were commonly used together in the GloVe's training corpus. For example:

Question 1	Where do I go in Germany?
Question 2	What are some good places to visit in France?
Human opinion	NOT DUPLICATE
Model's prediction	DUPLICATE

Here, the model incorrectly predicts them as duplicate. This is because France and Germany might have appeared in close contexts in the GloVe's training set, and GloVe thinks that both of those words have a strong connection, since they're both European countries.

This can be verified by computing the GloVe similarity between "France" and "Germany":

```
In [93]: nlp(u'France')[0].similarity(nlp(u'Germany')[0])

Out[93]: 0.68542368502653961
```

We can see a 68.5% similarity between the two words, hence this explains the problem that we had with our last question pair. This type of false positives are very detrimental to the user experience.

5.2 Reflection

This entire project was about supervised learning. We had a target variable, and some form of primitive features. However, being a Natural Language Processing task, features had to be extracted from the text first.