NLP-22 Detecting Semantically Similar Questions on Quora Dataset

December 14, 2018

Abstract

Websites like quora provide a platform for Question and Answering across various subjects and fields. However, the websites also contain loads of similar questions which have different sets of replies although they seek same answer. We aim to develop an application which can detect whether two questions on quora are semantically similar. More formally we want a model which when given two questions q_1 and q_2, gives

$$f(q_1, q_2) = \begin{cases} 0 & \text{if question can't be answered by the same answer} \\ 1 & \text{if question can be answered by the same answer} \end{cases}$$

depending on whether the questions are similar in terms of meaning or not. The criteria for two questions being semantically similar is that they seek same answer.[1]

1 Problem motivation

- **Widespread need of detecting semantic similarity**
  The problem of detecting semantic similarity is much easier than understanding semantics itself. The main reason for this is that we won't encounter the grounding problem while finding the semantic similarity or dissimilarity of two questions.
- **Need to eliminate duplicacies from QnA websites**
  If all similar questions can be routed to same page, then the answer-seekers can access all the answers provided for the question. To quote the quora article this would help, 'In order to build a high-quality knowledge base'. Solution to this problem can also be applied to other question answer forums like stackexchange which have a much more specific domain of questions but find a huge user base.

2 Implementation Details

2.1 Data

We are using the Quora datset released in January'16. The dataset consists of over 400,000 lines of potential question duplicate pairs. Each line contains IDs for each question in the pair, the full text for each question, and a binary value that indicates whether the line truly contains a duplicate pair.

However, the distribution of questions in the dataset is not representative of the distribution of questions asked on Quora. This is, because of the combination of sampling procedures and also due to some sanitization measures that have been applied to the final dataset (e.g. removal of questions with extremely long question details). Due to this reason we did not try to learn the word embeddings on this dataset and used standard word embeddings(or used alternate methods to learn them) for our approaches.

2.2 IBM Paper

In the paper 'Detecting Semantically Equivalent Questions in Online User Forums'[2], the authors describe an architecture of identifying duplicate questions. They use the *askubuntu* forum dataset. As the code for this paper is industrial, and so cannot be made public, we decided to implement it ourselves and make it available to the public.

2.2.1 Original Architecture

First the word embedding of each word is obtained by a learned matrix W^0.

$$r^w = W^0 v^0$$

where r^w is the dense vector representation of word and v^w is the one hot representation of the word. Then, for each word, a final representation is built by concatenating words in a window of size k around it in the question.

$$z_n = \left(r^{w_{n-(k-1)/2}}, ..., r^{w_{n+(k-1)/2}}\right)^\top$$

where z_n is the final representation of the word. This is followed by a linear and a non-linear transformation. The obtained vectors are then summed(over all the words in the question), and finally, another non-linearity is applied.

$$r_q = f\left(\sum_{n=1}^{N} \left[f(W^1 z_n + b^1)\right]\right)$$

where r_q is the vector representation of a question. After getting vectors for both the questions, r_{q_1} and r_{q_2}, the cosine similarity is computed.

$$s(q_1, q_2) = \frac{r_{q_1} \cdot r_{q_2}}{||r_{q_1}||||r_{q_2}||}$$

For training, SGD is used to minimize the square loss.

$$\theta = \sum_{(x,y)\in D} \frac{1}{2}(y - s_\theta(x))^2$$

2.2.2 Our implementation

As one of our main motives in this project was learning TensorFlow, we implemented this model using basic tesorflow (not using any on the top tool like keras, but basic TensorFlow). First, instead of learning the word embeddings from the data, we use Glove word vectors for finding dense vector representation of words. We did this for two reasons: (1) The corpus is small, (2) The model would become very data specific, and may over fit. The rest of the procedure is same for obtaining the question vectors. The similarity between questions is computed by first taking a cosine similarity, followed by passing it through a sigmoid, to squish it between 0 and 1.

$$s(q_1, q_2) = \sigma\left(\gamma \frac{r_{q_1} \cdot r_{q_2}}{||r_{q_1}||||r_{q_2}||}\right)$$

where γ is the scaling parameter for the cosine similarity. We use cross entropy loss for training.

$$\theta = -\sum_{(x,y)\in D} y \log s(x)$$

2.3 Erogol's Blog

In his blog[3], Eren Erogol describes his approach to solving the problem and suggests changes that can be made to enhance the accuracy of the model. We first implemented the baseline architecture and then attempted to tweak the network for better results. The approaches are described in this section.

2.3.1 Baseline Architecture

Constructing a Siamese network seems the most logical approach to solving the given problem. This involves finding an appropriate representation of each question followed by finding the 'distance' between them.

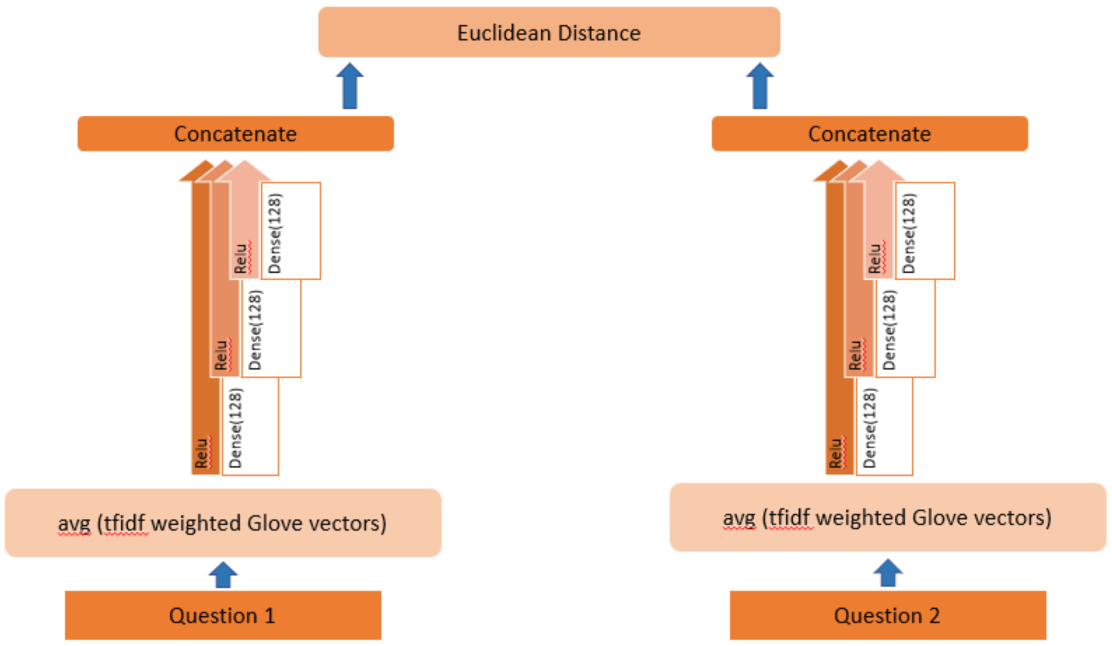

Figure 1: A schematic of the Baseline Architecture

To construct the feature vector for each question, first the average of Glove vectors (spacy model, vectors trained on Wikipedia Articles) of each word of the question is taken, weighted by its tfidf value. Following this, the newly constructed vector is passed into a network as described - The network consists of 3 dense layers of 128 nodes each and having Relu activation function. The output of one layer is successively passed onto the next layer after applying batch normalization. The final layer is constructed by concatenating the relu outputs of the three layers.

Upon obtaining the feature vectors of both questions, the euclidean distance between the two is calculated. If it is higher than a threshold value, the questions are classified as different, else as same.

The accuracy obtained for this architecture is : 80.21%

2.3.2 Further tweaks to the Baseline Architecture

1. **Preprocessing using BoW vectors**
   Instead of using averaged weighted Glove vectors, we used tf normalized vectors which were then passed into the network described previously. The accuracy for this approach was 55%.
   The motivation for using the BoW vectors was to capture the structure of the question and to compare

it to that of the other question as capturing the semantic meaning of question was proving to be difficult. We drew the conclusion that BoW vectors were not capturing the exact structure of the questions. The correct way to capture the structure of a question would be to look at its parse.

2. **Adding LSTM in the last layer**
   The architecture implemented was the following:
   The general structure is same as the Baseline structure. The main changes were made in the representation of the question vector. Instead of using the average of glove vectors, we used a Bidirectional LSTM. Each word in the question was converted into glove vector and this gives us a 2D matrix which is then passed to a bidirectional LSTM layer followed by the time distributed layer. The time distributed layer is basically a dense layer that acts on the output sequence of the Bidirectional LSTM layer and brings down the dimension of of the output of the LSTM layer. This returns back a 2D matrix which is passed through a 'Flatten' layer which concatenates the vectors giving a single dimensional vector. This vector is then passed in the same network as described above in the baseline architecture.
   The reasoning behind using a Bidirectional LSTM was to capture the relation between words in a question which would enable the model to learn the semantic meaning of the question.

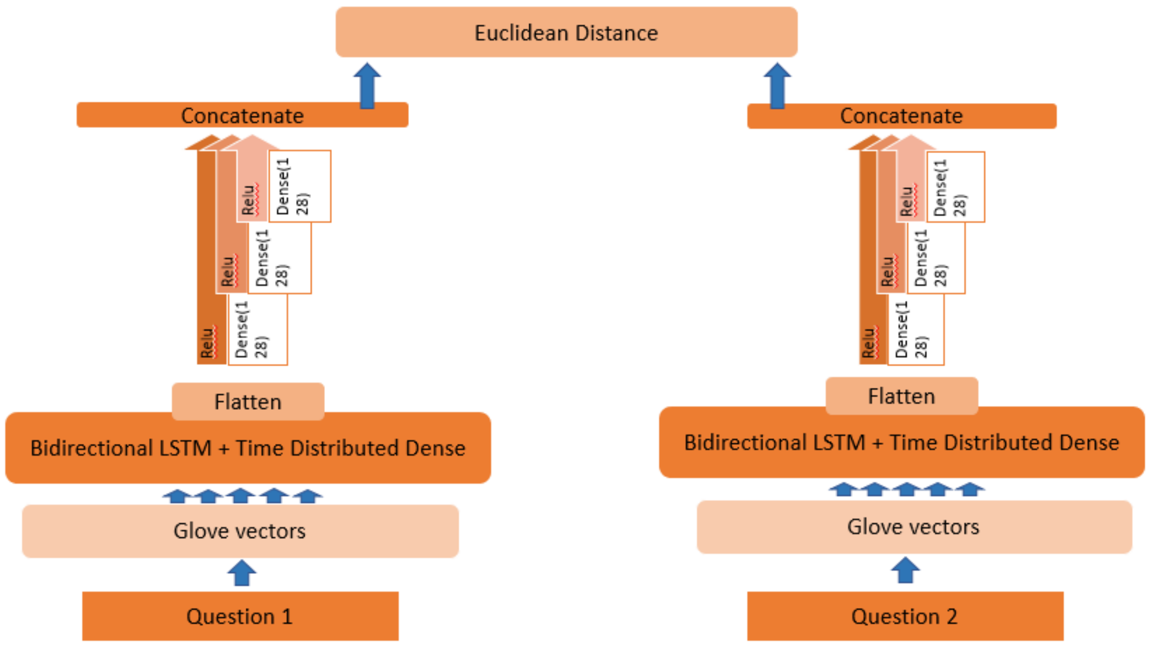

Figure 2: A schematic of the Baseline Architecture with additional LSTM layer

The accuracy achieved was 62.5% which is really low compared to the baseline structure. A possible reason for such lower accuracy is the lack of training data due to which the model isn't getting trained properly.

2.4 Adding Linguistic Constraints

Both in IBM Paper and Erogol, we tried to include some linguistic constraints. We analysed the cases that were in error and added linguistic constraints in order to imporve the performance. This was aimed to eliminate those cases which are being misclassified as being similar, however it seemed to us that simple constraints could help to eliminate these cases.These confusing cases were largely concentrated in the region near the threshold of decision, which motivated us further to give a negative confidence score in them. Given below are some of the methods we tried:

- Checking the **type of question**, which is decided by the question word. Apart from a few combinations, we can assume that the questions with different question words are different, atleast in terms of the answers they seek.
- **Matching the proper nouns/entities** verbatim in both the sentences, since they should match for the questions to be similar.

2.4.1 Problems with Linguistic Constraints

- **Type of question** We noticed that there are a large number of cases in the dataset which have been labelled to be similar even though they seem to seek different answers. We took independent comments from different people to confirm this. We analysed the number of mismatch of these question words in training set and the error set(in the test set) and found that they were present in similar proportions, which means that we will not benefit by including these constraints.
- **Proper Noun/ Entity matching**A problem with this approach is that we can have slightly different form of the proper noun in both the sentences, which may be clear to us due to our knowledge.
- **Exceptions to linguistic constraints**, it is clear that the constraints we used are not universal and there are many cases where they may fail. Questions starting from different question words may be same. For e.g.- Where do I get an Ice-cream? and What place do I get an Ice-cream? are similar. Also, proper nouns may exist in different forms due to abbreviations and alternate words.

In almost all the above discussed cases there was little or no improvement, the accuracy even decreased in some cases. The results for the case of IBM Research paper are tabulated in section 2.5

2.5 Experimental Results

2.5.1 IBM Paper Results

Hyper-parameters- We had kept two hyper-parameters while training the model.

- **Gamma**: This is the scaling factor for the cosine similarity between the two vector representations r_{q_1} and r_{q_2} of the questions.
- **Threshold**: The threshold between 0 and 1 which was used to assign a label to the two input questions. If the predicted output was greater than this threshold, the input questions were labeled as similar questions, and dissimilar otherwise.

The values of various scores achieved by using different values of the hyper-parameters are tabulated in Table 1 and Table 2

Hyper-parameters	Overall Accuracy(%)	Precision	Recall	F-measure

Gamma = 3
Threshold = 0.65
	66.86	0.594	0.333	0.427

Gamma = 3
Threshold = 0.7
	66.99	0.599	0.330	0.426

Gamma = 3
Threshold = 0.75
	67.06	0.604	0.322	0.420

Gamma = 4
Threshold = 0.65
	66.86	0.594	0.333	0.427

Gamma = 4
Threshold = 0.7
	66.99	0.599	0.330	0.426

Gamma = 4
Threshold = 0.75
	67.06	0.604	0.322	0.420

Gamma = 5
Threshold = 0.65
	66.86	0.594	0.333	0.427

Gamma = 5
Threshold = 0.7
	66.99	0.599	0.330	0.426

Gamma = 5
Threshold = 0.75
	67.06	0.604	0.322	0.420

Gamma = 6
Threshold = 0.65
	66.86	0.594	0.333	0.427

Gamma = 6
Threshold = 0.7
	66.99	0.599	0.330	0.426

Gamma = 6
Threshold = 0.75
	67.05	0.604	0.321	0.420

Gamma = 7
Threshold = 0.65
	66.85	0.594	0.333	0.427

Gamma = 7
Threshold = 0.7
	66.99	0.599	0.330	0.426

Gamma = 7
Threshold = 0.75
	67.06	0.604	0.322	0.420

Table 1: Experimental Results for IBM Paper without using Linguistic Constraints at Prediction time

Hyper-parameters	Overall Accuracy(%)	Precision	Recall	F-measure

Gamma = 3
Threshold = 0.65
	66.96	0.650	0.235	0.345

Gamma = 3
Threshold = 0.7
	66.34	0.669	0.182	0.286

Gamma = 3
Threshold = 0.75
	64.51	0.666	0.086	0.152

Gamma = 4
Threshold = 0.65
	66.96	0.650	0.235	0.345

Gamma = 4
Threshold = 0.7
	66.34	0.669	0.182	0.286

Gamma = 4
Threshold = 0.75
	64.51	0.666	0.086	0.152

Gamma = 5
Threshold = 0.65
	66.96	0.650	0.235	0.345

Gamma = 5
Threshold = 0.7
	66.34	0.669	0.182	0.286

Gamma = 5
Threshold = 0.75
	64.51	0.666	0.086	0.152

Gamma = 6
Threshold = 0.65
	66.95	0.650	0.235	0.345

Gamma = 6
Threshold = 0.7
	66.34	0.669	0.182	0.286

Gamma = 6
Threshold = 0.75
	64.51	0.666	0.086	0.286

Gamma = 7
Threshold = 0.65
	66.95	0.650	0.235	0.345

Gamma = 7
Threshold = 0.7
	66.34	0.669	0.182	0.286

Gamma = 7
Threshold = 0.75
	64.51	0.666	0.085	0.182

Table 2: Experimental Results for IBM Paper along with Linguistic Constraints at Prediction time

2.5.2 Erogol Results

MODEL	ACCURACY
Baseline Architecture(proposed)	79%
Baseline Architecture(tweaked)	80.21%
Bi-Directional LSTM	62.5%

2.6 Using the Word Hierarchy

We had proposed that we can ignore the presence of a word from one of the sentences given that it's specific form is present in both the sentences. Q1 "How to open a Bakery business?", Q2 "How do I start a Bakery?" and Q3 "How to start a business?" Q1 and 2 are similar while 1 and 3 are not, this tells us that Bakery is more specific form of Business. Therefore, if we match a word which is lower in hierarchy or is more specific(Bakery), then the general words(Business) higher up can be neglected. We tried to implement this but the biggest problem we faced was that the existing set of hypernyms and hyponyms are insufficient to be able to achieve what we want. The problem was that our definition of hyponyms was very specific to the type of question, which could not be captured using wordnet. Therefore, this approach could not give us considerably good results.

2.7 Conclusion

It seems that in order to find similarity between two sentences, we won't encounter the grounding problem. However, after analysing the errors made in the currently employed methods, we felt that semantic based approaches should be used. Almost all of the current metohds are based on findind similarity in the sentence structure and do not try to use the semantics for the task. This brute force matching of structure leads to a lot of false positives, where the sentences seem to be of the same form.

References

[1] "Semantic question matching with deep learning quora blog." `https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning`.

[2] D. Bogdanova, C. N. dos Santos, L. Barbosa, and B. Zadrozny, "Detecting semantically equivalent questions in online user forums," in *CoNLL*, pp. 123–131, ACL, 2015.

[3] E. Golge, "Duplicate question detection with deep learning on quora dataset." `http://www.erogol.com/duplicate-question-detection-deep-learning/`.

3 Work Distribution

Almost everybody contributed equally.

Name	%Contribution
Abhishek Kumar	17%
Bhuvi Gupta	17%
Manish Kumar Bera	17%
Mohd Abbas Zaidi	16%
Mrinal Dogra	16%
Prann Bansal	17%