Quora Question Pairs Similarity Prediction

Team:
Rujitha Vennugopalan rujitha2@illinois.edu
Senthil Ramachandran skr6@illinois.edu

Abstract

We explored two approaches based on Long Short-Term Memory (LSTM) networks on the Quora duplicate question dataset. The first model uses a Siamese architecture with the learned representations from a single LSTM running on both sentences. The second method uses two LSTMs with the two sentences in sequence, and the second attending on the first (word-by-word attention). Our best model achieved 79.5% F_1 with 84.3% accuracy on the test set.

1 Introduction

Understanding semantic relatedness of sentences would allow understanding of much of the user-generated content on the internet, such as on Quora. In this paper we address the problem of actual duplication or exact semantic coincidence between questions. Solving this problem would be useful to helping Quora organize and deduplicate their knowledge base.

The pairs of questions in our problem will be already similar in that they have many low document-frequency words in common; however, the negative examples will have subtle semantic differences. Some of the differences are due to different scopes of each question. For example, a question asking why something happens is different for a question asking whether that thing happens. Conversely, our model needs to recognize when two questions use different words and phrases with the same semantic meaning, or the users' intended questions are the same and would elicit the same answers. Our model tries to learn these patterns.

1.1 Data

The Quora duplicate questions public dataset contains 404k pairs of Quora questions.[1] In our experiments we excluded pairs with non-ASCII characters. We split the data randomly into 243k train examples, 80k dev examples, and 80k test examples. To validate the dataset's labels, we did a blind test on 200 randomly sampled instances to see how well an independent person's judgements would agree with the given labels. For preprocessing, we used the tokenizer in the nltk Python package.

2 Background/Related Work

Previous work on semantic relatedness of sentences has focused on logical inference and entailment through based on the Stanford Natural Language Inference Corpus [2]. The first of these papers which focused on attention methods using LSTMs was Rocktaschel et al. (2015) [1] which introduced word-by-word attention methods with the hypothesis attending on the premise. An important difference between our problem and the entailment problem is that our problem is posed with an inherent symmetry between the two sentences. We limit our approaches to symmetric ones, including running pairs twice in the sequence-to-sequence model with one-sided attention.

[1]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
[2]https://nlp.stanford.edu/projects/snli/

3 Method or Approach

3.1 Machine Learning

3.1.1 Logistic Regression

A very simple approach to detecting similarity between a pair of questions would be to look at unique words in the first question that are also present in the second question as a ratio of the total words in both questions. This number could then be used in a simple model such as logistic regression to predict duplicate versus different questions.

3.1.2 Hyperparameter Tuning

Cross-validation is a good technique to tune model parameters like regularization factor and the tolerance for stopping criteria (for determining when to stop training). Here, a validation set is held out from the training data for each run (called fold) while the model is trained on the remaining training data and then evaluated on the validation set. This is repeated for the total number of folds (say five or 10) and the parameters from the fold with the best evaluation score are used as the optimum parameters for the model.

3.1.3 SVM

SVM (Support Vector Machine) refers to a kernel based machine learning algorithm where a vector space is mapped into another vector space where training examples are linearly separable, and two hyper-planes corresponding to two classes are defined with the maximal margin between them as boundaries of classification. We proposed a SVM-based solution to compute the semantic similarity between two sentences

3.1.4 Tree Based Model

Tree based models often give excellent results and are frequently applied in practice. Quora for example currently uses a random forest with hand engineered features on this problem (Dandekar, 2017). For this reason we were interested in applying these models to this problem to understand how they performed and how they differed from linear models and neural networks.

3.2 Deep Learning

3.2.1 Embeddings

We used GloVe pre-trained word vectors to initialize our word embeddings.3 After experimenting with different dimensions, we found the 300-dimensional vectors worked best for our problem. We initialized training words not contained in GloVe to small random vectors. We experimented with both fixed and trainable embeddings.

3.2.2 Bag of words approach

As a baseline, we used methods based on the mean and elementwise-maximum of embeddings of words in each sentence. We tried one- and two-layer Siamese networks which take the weighted inner product of these vectors between the two sentences. We also added Euclidean distance and element-wise product as features.

3.2.3 Siamese LSTM approach

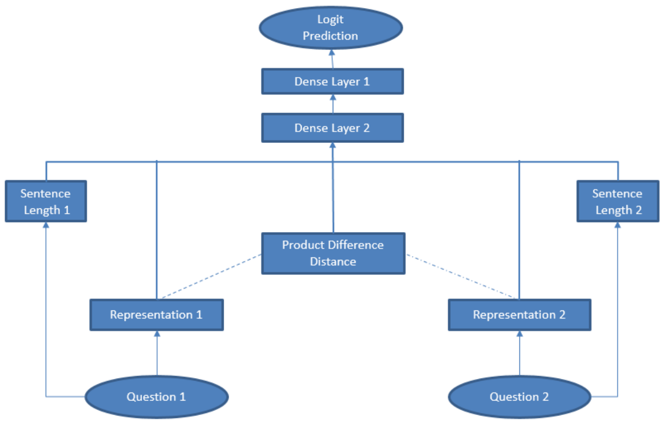
Let L be the maximum sentence length, and H be the hidden size. We use a single LSTM to produce output vectors at the sentence level, $h_L^{(1)}$ and $h_L^{(2)}$, from the word-level embeddings for the first L words in each sentence. We then add a sentence length feature to each h_L, and take Hadamard product, squared difference, squared Euclidean distance, as well as the original $h_L^{(1)}$ and $h_L^{(2)}$, and feed it into one or two feed-forward layers which go to a softmax for the prediction. The LSTM cells are implemented with the standard equations for the gates, where x_t are the word embeddings at the t-th word

$$i_t = \sigma\left(w^{(i)}x_t + U^{(i)}h_{t-1}\right)$$
$$f_t = \sigma\left(w^{(f)}x_t + U^{(f)}h_{t-1}\right)$$
$$o_t = \sigma\left(w^{(o)}x_t + U^{(o)}h_{t-1}\right)$$
$$f_t = \sigma\left(w^{(f)}x_t + U^{(f)}h_{t-1}\right)$$
$$\tilde{c}_t = tanh\left(w^{(c)}x^t + U^{(c)}h_{t-1}\right)$$
$$h_t = o_t \circ tanh(c_t)$$

Here $\sigma(x)$ is the sigmoid function, h_0 is initialized to be zero, and the parameters $w^{(i)}; w^{(f)}; w^{(o)}; w^{(c)} \in \mathbb{R}^{H*k}$ and $U^{(i)}; U^{(f)}\ ;\ U^{(o)}; U^{(c)} \in \mathbb{R}^{H*H}$, where κ is the embedding size.

Figure 1: Siamese architecture. The same LSTM is used to produce the two representations.

https://nlp.stanford.edu/projects/glove/



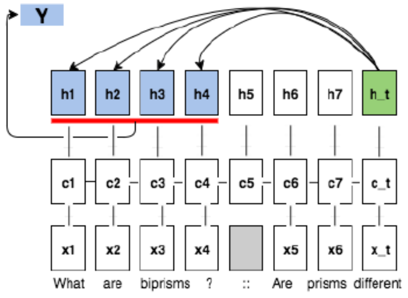
3.2.4 Sequence-to-sequence Word-by-word Attention

The sequence-to-sequence model uses two LSTMs (with separate parameters) where the last state of the first LSTM is passed to the second. Running on our pairs of questions, we made this model symmetric by running the sequence to sequence in both orderings of the pair and averaging the h^* from each output.
For the attention mechanism, we have, as in Rocktaschel [1] (11)-(14)

$$\alpha_t = softmax\big(score(Y, h_t)\big)$$
$$r_t \;= Y\, \alpha_t^T + \tanh(W_{r_{t-1}}^t)$$
$$h^* = \sigma\big(w_{\gamma L}^P + \omega^{\varkappa} h_N\big)$$
$$\tilde{y} = softmax(w_h^* + b)$$

Here $\mathrm{Y} \in \mathbb{H} * \mathbb{L}$ is the concatenation of the output vectors of the first sentence, $r_t \in \mathbb{R}^k$ where r_0 is initialized to zero, the parameters $W_y; W_h;\ W_r;\ W_t;\ W_p;\ W_x\ \mathbb{R}^{H*H}$. The final line represents the feed-forward of h^*; this may be instead two or more dense layers.

Figure 2: Attention scoring mechanism of a single word over all words in first sentence



The model in [1] uses a scoring that is additive between Y and h_t (2a). We also considered a bilinear scoring type to try to include interactions between Y and h_t (2b). Here $e_l \in \mathbb{R}^l$ is a vector of 1's to broadcast the matrix it is multiplying through outer product.

$$score(Y, h_t) = \begin{cases} w^T \tanh(W^y Y + (W^h h_t + W^r r_{t-1}) \otimes e_L) & \text{(2a)} \\ Y^T((W^h h_t + W^r r_{t-1}) \otimes e_L) & \text{(2b)} \\ [Y; X_1]^T((W^h[[h_t; x_t] + W^r r_{t-1}) \otimes e_L) & \text{(2c)} \end{cases}$$

The motivation behind the third scoring type (2c) was that adding the original embeddings would have a kind of regularizing effect through fresh, more objective and less contextualized representations, or to bring up concepts that the LSTM may have overlooked. We thought it would encourage the attention mechanism to match words that are synonymous since you essentially inner product together the original embeddings, and this would be favorable based on the visualizations of the learned attention scoring in [1] (page 7).

3.2.5 Loss functions

We use the cross-entropy loss over a softmax as a two-class classification problem, minimizing

$$J = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i).$$

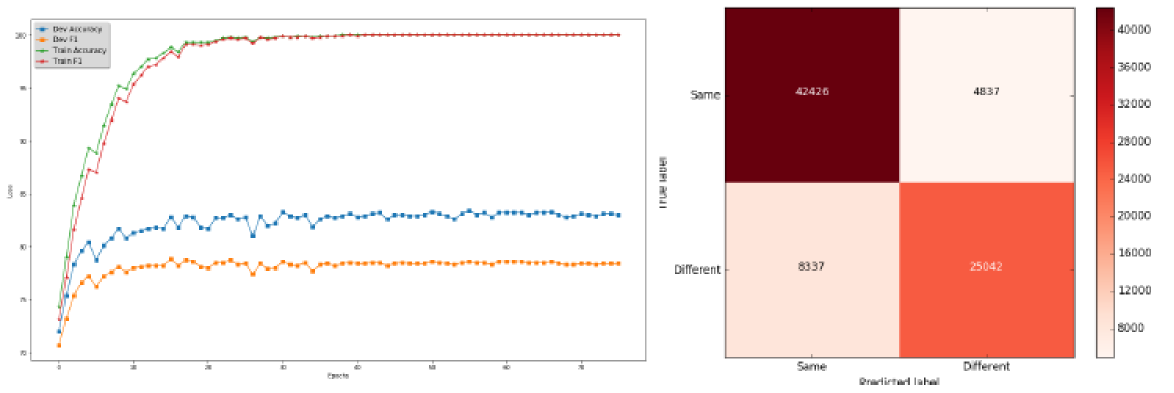
We also considered a margin-based loss.

4 Experiments

Figure 3: Summary of main results

		Dev				**Test**			
Model	$\|\theta\|$	Precision	Recall	F1	Accuracy	Precision	Recall	F1	accuracy
Siamese with bag of words	602k	65.2	83.7	73.3	77.4	64.7	84.2	73.2	77.3
Siamese with LSTM	8.8M	72.8	87.2	79.3	84.3	73.0	86.8	79.3	84.2
Seq2Seq LSTM with Attention	381K	70.1	84.1	76.5	80.9	70.2	83.7	76.4	80.8
Ensemble	9.2M	74.6	85.2	79.5	83.9	74.9	84.7	79.5	83.8
Human (200 examples)	-	67.2	80.4	73.2	82.5	-	-	-	-

Figure 4: Example training process on a Siamese LSTM model (left) and confusion matrix on test data (right)



4.1 Hidden size / parameter size

In our explorations, we found the Siamsese LSTM performed well with a hidden size of about H = 1000. It would not be possible to train the Seq2Seq model with word-by-word attention on a hidden size that large; in this model we used a maximum hidden size of H = 256. In some sense the Seq2Seq model does reasonably well relative to its parameter size.

4.2 Ensemble

We ensembled the best performing single model of the Siamese LSTM with the best single model of Seq2Seq with Attention. This gave us a slight lift over the best Siamese LSTM even though the Attention model performs significantly worse alone.

4.3 Training details

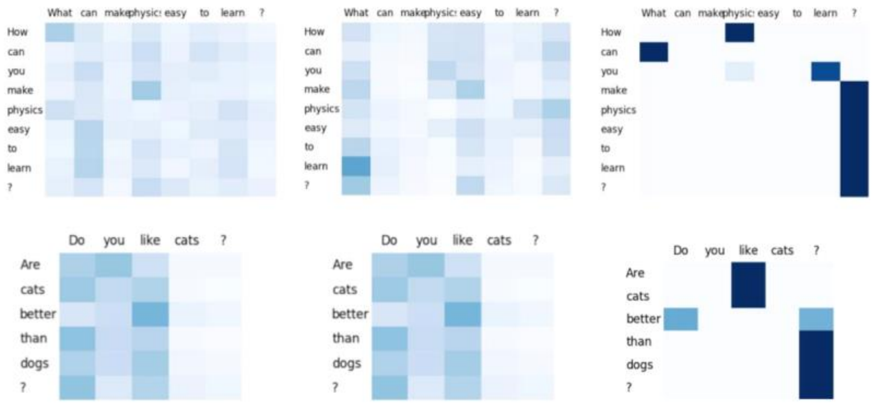
For speed of development we used L=35 based on the distribution of sentence length; however, all methods would perform better with a higher maximum length (the median sentence length is 10-11 words, and the maximum is around 120 words). We used TensorFlow's dynamic rnn to be able to unroll the LSTM by sentence length within batches. We found a staircase learning rate with decay factor of 0.9 worked best. We did masking in by multiplying elementwise by the binary mask, and in attention by adding the logarithm of the binary mask to $score(Y, h_t)$ We used the AdamOptimizer implemented in tensorflow and tried a different learning rates.

4.4 Dropout and regularization

We used the TensorFlow Dropout Wrapper on the LSTM cells. We found it is very important not to use dropout on the $h^{(1)}$ and $h^{(2)}$ vectors independently since we are looking at their similarity and doing so would distort the similarity signal. Since we did not use dropout on the h activations, we used l_2 l2 regularization on the W parameter. On the Siamese LSTM model, we did a small grid search on the Dev set over dropout [0.7, 0.8, 0.95], _ [0.1, 0.5, 0.2], and learning rate [$3e^{-4}$, $6e^{-4}$, $1e^{-3}$]; however, the model's major parameter was the hidden size.

4.4.1 Attention word-by-word patterns

Figure 5: Trained word by word attention scoring on example sentences using different scoring functions. The 1st column is additive scoring (2a); 2nd column is bilinear scoring (2b); 3rd column is bilinear adding the original embeddings (2c).



These patterns are perplexing for us in comparison to the Rocktaschel paper [1], and it is possible that the model did not train the word-by-word scoring weights in any conceptual and generalizable way. However, it is not necessarily true that these patterns are expected to be the same as the entailment word-by-word patterns in [1] where the model found high attention on pairs of words that were similar or highly related. In our problem, our pairs of questions are already sampled to have matching words, and the problem over this distribution is more focused on difference than similarity. For example, the difference between starting a question with "how" versus "why" is critical and it would make sense for later words in the question to focus on these first words.
At the very least, adding attention on the individual word outputs ($h_t's$) of the first LSTM on the first sentence is better than just taking the last cell's output and passing it into the next LSTM and taking the output from the last cell of the second LSTM. We validated that sequence-to-sequence models with any of the three attention methods perform better than without any attention.

Figure 6: Attention methods scoring detail

Score Type	Dev Precision	Dev Recall	Dev F1	Dev accuracy
(2a)	70.0	80.0	74.7	80.1
(2b)	69.9	83.9	76.3	80.8
(2c)	70.1	84.1	76.5	80.9

The third scoring function (2c) seemed to do as well or slightly better, though for different reasons than intended.
A possible interpretation is that this scoring function had a kind of regularizing effect where the model mostly pays attention to the question mark, where the whole sentence's meaning is resolved, and a little to earlier important resolution points in the sentence. The earlier parts of sentence focus on earlier parts of sentence, and the end focuses on final h_t which is entire sentence processed. In this way it becomes an intermediate model between a sequence-to-sequence model without attention or a siamese model which only takes the last cell memory or output from the first sentence, and full pairwise word to word attention.

4.5 Qualitative analysis
We looked at the pairs in the Dev set which contributed most to the loss (positive examples with the lowest predicted logits and negative examples with the highest predicted logits).

Selected most strongly mis predicted negative examples (false positives)
N.1.a. How does weight loss transformation affect your personality?
N.1.b. Did your weight loss transformation affect your personality?
N.2.a. Can Aam Aadmi Party win the 2015 Delhi election ?
N.2.b. What are the possibilities of Aam Aadmi Party winning the Delhi elections ?
N.3.a. How should I start writing on Quora ?
N.3.b. How should I start writing answers on Quora ?

Selected most strongly mis predicted positive examples (false negatives)
P.1.a. Which is the best coaching institute for the GMAT ?
P.1.b. Which is the best coaching for GMAT ? Is it Byju's, Jamboree , Manya Princeton , Pythagurus , CrackVerbal , Meritnation or EduShastra ?
P.2.a. Why do you use a capo d'astro ?
P.2.b. When does a guitarist need to use a capo dastro ?
We believe some of these negative examples ([N1-N3]) may be mislabeled or ambiguous, which is why our model was so confident they were positive. In the positive examples, the questions require a lot of background knowledge to understand their relationship. For example in [P2], you need to know that capo d'astro is only or primarily used by guitarist, thus making the scopes of the questions the same.

Selected correctly predicted positive examples (true positives)
P.5.a. Will the Supreme Court's decision of playing the National Anthem before movie screenings affect your patriotism?
P.5.b. What's your stand on the recent Supreme Court's order about national anthem in cinema halls?
P.6.a. Is our universe just a computer simulation ?
P.6.b. Could our universe actually be a computer program ?
Figure 10: Selected correctly predicted negative examples (true negatives)
N.5.a. How many two-digit numbers can you form using the digits 1,2,3,4,5,6,7,8 and 9 without repetition?
N.5.b How many two-digit numbers can you form using the digits 1,2,5,7,8 and 9 without repetition?

In these correctly predicted examples, [P5] needed to recognize that the two words "decision" and "order", as well as "movie" and "cinema" were semantically the same or very similar—or at least that they were not so different that they would make the two sentences divergent in meaning. This depends on the word embeddings, trained or untrained, and also the LSTM(s) to process the sequence. Similarly in [P6], the model needed to recognize that "program" had similar meaning as "simulation". In the negative example [N5], as the LSTM needs to process a sequence of numbers, the output of a sequence of 10 consecutive numbers in N.5.a. must have been different enough from the output of a sequence of 6 consecutive numbers.

4.6 Out-of-vocabulary handling
We loaded the GloVe vectors for all of the Train, Dev, and Test sets in our dictionary; the words that were not in Train were not touched during training time. We realized at the last minute that we were not training the Unk vector since we put all of our training words in the vocabulary. We tried assigning words to the Unk vector randomly with a 1% probability in training, so the Unk vector can pick up a good value that fits in the embedding distribution—this would act as a kind of dropout over the vocabulary. We looked at the performance on pairs of questions containing out-of-vocabulary words versus overall. These statistics are based on a bag of words model.

	Dev precision	Dev recall	Dev F1	Number Instance
Overall	64.7	84.2	73.2	80k
Having out of vocabulary words	69.8	59.8	64.4	14k

As expected, the model performs worse on pairs with out-of-vocabulary words because it does not have the embeddings to fit to those words and can only extract a generic signal from them.

4.6.1 Analysis by sentence length

We took relative sentence length as the percentage difference in number of words from the longer of the two sentences. These performance statistics are based on an early version of the siamese LSTM model.

	Dev precision	Dev recall	Dev F1	Number Instance
Overall	75.0	81.4	78.5	80k
Relative length <20%	76.1	83.4	79.6	39k
20% <Relative length <40%	75.4	79.3	77.3	23k
Relative length >40%	74.7	74.6	74.7	15k

This breakdown shows a significant decline in performance on pairs with higher relative length difference. It is likely that pairs with high relative length difference will tend to be inherently harder to predict correctly; at the same time, length differences of 2x should probably not be an issue in sentence embeddings produced from an LSTM which should encapsulate the semantic meaning of the sentence; thus this information gives us an area for improvement.

4.6.2 Blind study findings
Our blind study on 200 sampled rows of the dev set found 83.5% match rate with the dataset's labels (see last row of [3]). There is a caveat that we were not given the same instructions as the original judges. The F_1 score was fairly centered, with recall a bit higher than precision. In doing this test, we found a couple inconsistencies with the labeling. When one question's subject is a principal component or identifying characteristic of the other question's subject, it is not clear whether the correct label should be True or False. For example, the pair

P.11.a. Were the Clintons paid by the Clinton Foundation?
P.11.b. Was Chelsea Clinton really paid $700,000 a year working for the Clinton Foundation ?
is marked Positive for being the same, but the pair

N.11.a. What does follow mean on Quora ?
N.11.b. What do “upvote”, “downvote”, “follow” a question mean on Quora ?
is marked Negative.

We think this difference in labels shows an inconsistency since the two pairs are related in a similar way. Chelsea Clinton is a member of the Clinton family and the [P.11.b] is asking about a specific amount she was paid, but this specific occurrence is identifying and characteristic of the concept in [P.11.a]. In [N.11.a], the “follow” button and is a principal component of the topic in [N.11.b] (all three buttons), and any answer to [N.11.b] would contain an answer to [N.11.a].

There are also a few pairs that directly contradict other pairs. For example

P.12.a. What was the signi_cance of the battle of Somme, and how did this battle compare and contrast to the Battle of Riyadh?
P.12.b. What was the signi_cance of the battle of Somme, and how did this battle compare and contrast to the Battle of France?
N.12.a. What was the signi_cance of the battle of Somme, and how did this battle compare and contrast to the Battle of Taiyuan?
N.12.b. What was the signi_cance of the battle of Somme, and how did this battle compare and contrast to the Battle of Eslands River?

The first pair is labeled as positive and the second pair is labeled as negative, even though the pairs' relationships are identical. (These may have been included intentionally to test the crowdsourcing—there are about 235 pairs exactly like this example with varying labels.)

4.7 Comparison to Quora's results and further development

On one hand, we believe our results are not directly comparable to Quora's result of 88% F_1 and 87% accuracy [11] in that we did not use the same embeddings and did not do the same preprocessing. Their results were based on word embeddings trained on a corpus of Quora data. We believe that a significant part of the gap in results (though not at all close to all of it) may be due to a difference in embeddings and lack of coverage of our pretrained embeddings. Also, based on our blind study and our experience so far with the data, their results may be at the very limit of what is possible on this dataset and their models might be highly specialized to predict on Quora platform questions (versus general questions of a certain length). Since user-inputed language on an online platform is more free and has more variations than standard English, or the language in Wikipedia (which GloVe was trained on), we feel our results can be greatly improved fundamentally by using character-based models, engineering overlap features, as well as minor developments such as using features like Named Entity tags and attempting spelling corrections. We might also use separate models for high string similarity cases and large relative-length difference cases; this may work better in practice though theoretically it should not be necessary. In addition to improving performance, these implementations would give a very new perspective to the problem.

5 Conclusion

In this project, we produced competitive results on the Quora duplicate dataset problem using two basic architectures with LSTMs, and played with scoring methods in the attention mechanism. In the given time we reached 79.5% F_1 and 83.8% accuracy. The Siamese LSTM architecture (with an order of magnitude more parameters) outperformed the sequence-to-sequence with attention methods, though the attention methods were also well above the bag of words baseline. We are still actively developing our methods and look forward to improving our performance further.

Appendix:- Team Member contribution:
Rujitha Vennugopalan:-

1. Collected dataset for Quora duplicate prediction
2. Exploratory Data Analysis
    1. Distribution of data points among output classes
    2. Number of occurrences of each question
    3. Basic Feature Extraction (before cleaning)
    4. freq_qid1: Number of times the question 1 repeated in the whole data set.
    5. freq_qid2: Number of times the question 2 repeated in the whole data set.
    6. q1len: Length of the question1
    7. q2len: Length of the question2
    8. q1_n_words: Number of words in question 1
    9. q2_n_words: Number of words in question 2
    10. word_common: Number of common words in question 1 and question.
    11. word_total: Total number of words in question 1 and question 2.
    12. word_share: This is equal to (word_common/word_total).
    13. freq1+freq2: Sum of the frequencies of question 1 and question 2.
    14. freq1-freq2: Difference in frequencies of question 1 and question 2.
3. Analysis of some of the extracted features
4. Preprocessing of Text
5. Building Random Model
6. TFIDF and IDF
7. Linear Regression Model
8. Linear Regression Hyper Parameter tuning

Senthil Ramachandran

1. Advanced Feature Extraction (NLP and Fuzzy Features)
2. Definition:
    a. Token: You get a token by splitting sentence a space
    b. Stop_Word : stop words as per NLTK.
    c. Word : A token that is not a stop_word
3. Features:
    a. cwc_min : Ratio of common_word_count to min length of word count of Q1 and Q2
    b. cwc_min = common_word_count / (min(len(q1_words), len(q2_words))
    c. cwc_max : Ratio of common_word_count to max length of word count of Q1 and Q2
    d. cwc_max = common_word_count / (max(len(q1_words), len(q2_words))
    e. csc_min : Ratio of common_stop_count to min length of stop count of Q1 and Q2
    f. csc_min = common_stop_count / (min(len(q1_stops), len(q2_stops))
    g. csc_max : Ratio of common_stop_count to max length of stop count of Q1 and Q2
    h. csc_max = common_stop_count / (max(len(q1_stops), len(q2_stops))
    i. ctc_min : Ratio of common_token_count to min length of token count of Q1 and Q2
    j. ctc_min = common_token_count / (min(len(q1_tokens), len(q2_tokens))
    k. ctc_max : Ratio of common_token_count to max length of token count of Q1 and Q2
    l. ctc_max = common_token_count / (max(len(q1_tokens), len(q2_tokens))
    m. last_word_eq : Check if First word of both questions is equal or not
    n. last_word_eq = int(q1_tokens[-1] == q2_tokens[-1])
    o. first_word_eq : Check if First word of both questions is equal or not
    p. first_word_eq = int(q1_tokens[0] == q2_tokens[0])
    q. abs_len_diff : Abs. length difference
    r. abs_len_diff = abs(len(q1_tokens) - len(q2_tokens))
    s. mean_len : Average Token Length of both Questions
    t. mean_len = (len(q1_tokens) + len(q2_tokens))/2
    u. fuzz ratio
    v. fuzz partial ratio - This calculates how similar the sentences are partially.
    w. token sort ratio - The token sort ratio is like tokenizing sentence and joining tokens alphabetically into a string and then checking the order similarity
    x. token set ratio - token sort ratio would not help if the sentences are of lengths that largely differ but having some sub sentences meaning similar. So to solve this we use token_set_ratio.
4. SVM Model
5. SVM Hyper Parameter tuning.
6. Confusion Metrics & Visualization
7. Precision, Recall, F1 Score and Accuracy

Combined Team work:

1. Analysis of extracted features
2. Futurizing text data with tfidf weighted word-vectors
3. Random Forest Model
4. Bag of Words
5. Embeddings
6. Siamese LSTM approach
7. Sequence-to-sequence Word-by-word Attention
8. Attention word-by-word patterns
9. Qualitative analysis
10. Out-of-vocabulary handling
11. Analysis by sentence length
12. Project Report Creation

References

[1] Tim Rocktaschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, Phil Blunsom. Reasoning about entailment with neural attention. In ICLR 2016.
[2] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In EMNLP, 2015.
[3] Jonas Mueller and Aditya Thyagarajan. Siamese Recurrent Architectures for Learning Sentence Similarity. In AAAI, 2016.
[4] Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. In Proceedings of NAACL, 2016.
[5] Gabor Angeli and Christopher D. Manning. Naturalli: Natural logic inference for common sense reasoning. In EMNLP, 2014.
[6] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In EMNLP, 2015.
[7] Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, Wojciech Zaremba. Addressing the Rare Word Problem in Neural Machine Translation. In ACL, 2015.
[8] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In NIPS, 2014.
[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.
[10] Jiang Zhao, Tian Tian Zhu, and Man Lan. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In SemEval, 2014.
Nikhil Dandekar. 2017. Semantic question matching with deep learning. ”https://engineering.quora.com/Semantic-QuestionMatching-with-Deep-Learning”. Chapman Siu. 2016. Duplicate question detection using online learning

Blog posts

[11] Lili Jiang, Shuo Chang, and Nikhil Dandekar. (2017, Feb 13). Semantic Question Matching with Deep Learning [Blog post]. Retrieved from https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning
[12] Matthew Honnibal. (2017, Feb 13). Deep text-pair classification with Quora’s 2017 question dataset [Blog post]. Retrieved from https://explosion.ai/blog/quora-deep-text-pair-classification, fuzzywuzzy from https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/