Improving Semantic Question Matching of Quora Dataset by Feature-rich Classifiers and LSTM Encoders

Tian Chen
Department of Statistics
University of California, Irvine
tianc2@uci.edu

Huan Chen
School of Information and Computer Science
University of California, Irvine
huanc3@uci.edu

Abstract

In this paper, we are interested in identifying duplicate questions for a Quora question pair dataset. Previous work are mostly developed on small dataset (e.g. 5k pairs). We thus propose to use two data-intensive models for this task: a feature-rich classifier model and a neural network model using LSTM sentence embedding. They are both proposed by Bowman (Bowman et al., 2015) and are developed on a large corpus: Stanford Natural Language Inference (SNLI). We evaluate these two methods on a subset of Quora's dataset and showed that they both outperforms the baseline method. Overall, the feature-rich classifier based on lexicalized features achieves the best performance.

1 Introduction

Recently, Quora released over 400,000 pairs of potential duplicate questions on Kaggle and competitors are required to identify the actual duplicate pairs. That being said, given two sentences, we have to determine if they have the same meaning. This is the problem of paraphrase identification. In this paper, we name the task as semantic question matching, since the sentences are all questions. This is an important NLP task in many applications. As for on-line Question-Answering (QA) communities such as Quora and Stack-Overflow, it helps to maintain their high quality content. It also helps prevent spam users from duplicating the questions and answers from existing ones to gain reputation. In terms of information retrieval, with text-pair identification, query refinement can be conducted to perform targeted searching, etc.

In the Quora dataset, a sentence pair of the input data set of this task can be ”How do I read and find my YouTube comments?” and ”How can I see all my Youtube comments?”. The expected output should be 1, which means duplicate. Another example is ”What's causing someone to be jealous?”, ”What can I do to avoid being jealous of someone?”. The expected output should be 0, which means they are not duplicate.

Many techniques (Fernando and Stevenson, 2008; Ji and Eisenstein, 2013; Cheng and Kartsaklis, 2015) for paraphrase identification are developed on relative small dataset (of size 5K), which are limited and difficult to extend to data-intensive applications. Recent competitive methods are mostly developed on a large corpus: the Stanford Natural Language Inference (SNLI) corpus, which is released in 2015. In particular, (Bowman et al., 2015) proposed to use a feature-rich classifier and LSTM encoder with neural network model, which achieved comparable performance for SNLI. The size of Quora's dataset (400k pairs) is similar to SNLI (570k pairs), but since Quora's data are actual questions, it potentially will be more noisy and challenging.

In this paper, we plan to apply feature-rich models (lexicalized features such as cross bigram) and LSTM encoder (Bowman et al., 2015) to Quora's dataset. The baseline method we used here is a feature-based random forest model, which has been implemented by the engineering group at Quora (Dandekar, 2017).

We evaluated our method on a subset (8,000 pairs) of Quora's dataset. Overall the cross-unigram model achieves highest accuracy (dev: 0.68, test: 0.69). Regarding the LSTM encoder model, it somehow undergoes the problem of overfitting using Bowman's method. However, the LSTM embedding model still outperforms the baseline model (sum of words embedding) for development data. We also showed that LSTM generates better sentence representation.

2 Related Work

Early work for paraphrase identification includes techniques such as using semantic similarity via Wordnet. For example, (Fernando and Stevenson, 2008) used wordnet to calculate a matrix similarity between two sentences. Another famous and competitive method is developed by Socher (Socher et al., 2011), which is based on recursive autoencoders with a dynamic pooling layer. There is also a rencent trend in applying deep learning on top of word embeddings (Bogdanova et al., 2015). However, all these techniques are evaluated on small dataset, such as Microsoft Research Paraphrase Corpus which contains only 5,000 sentence pairs, which is too small for developing data-intensive and wide-coverage models(Bowman et al., 2015). Recent state-of-art methods for paraphrase identification are developed on the Stanford Natural Language Inference (SNLI) corpus, which provides 570K pairs of sentences. Based on this corpus, one branch of research is developing sentence encoding-based models and other neural network models. For example, (Bowman et al., 2015) proposed to use 100d LSTM encoders for sentence embedding and also compared it to some feature-rich classifiers.

The questions pairs data provided by Quora has 400K pairs and is of similar magnitdue to the SNLI corpus. The advantage of Quora dataset over SNLI corpus is that it's less artificial. The sentences and labels in SNLI is created by humans (Mechanical Turk contributors). The vocabularies are quite limited and sentences are literal(Honnibal, 2017). While the Quora dataset is from real world, which will be more challenging to solve compared to the SNLI dataset.

3 Approach

3.1 Feature-rich classifier

Quora's engineering group proposed to use some basic features such as similarity of the average of the word2vec embeddings of tokens, the number of common words, the number of common topics labeled on the questions, and the POS tags of the words. (Dandekar, 2017).

Given two sentences, naive approaches to label them include stripping the stop words and calculating similarities (Cosine or Jaccard) between the remaining tokens, however, it does not consider the semantic information. Word embeddings can be used to calculate the semantic similarity based on the matrix representations of the two sentences. Linear machine learning models can also be used to classify the sentence pairs, however, the features matter, simple features fail to achieve good performance. To address the challenge here, considering both the semantic and lexical information of the input sentences, we proposed to use some sentence-level features (lexical and semantic):

- Cross-unigrams: an indicator feature over the unigram of the hypothesis who shares the same POS tag with the one in the premise
- Cross-bigrams: an indicator feature over the bigram of the hypothesis who shares the same POS tag with the one in the premise
- Reverse bigrams: an indicator feature over the unigram over the hypothesis who shares the same POS tag with the sequential one in the premise
- lexical mapping: some acronyms and entities can be matched by external known mappings, such as people, organization, locations, authors and books, singers and songs, etc.

3.2 LSTM sentence embedding

Framework The focus of this method is to generate more informative sentence representation/embedding by LSTM, thus we will not consider strong neural network models. We first performed word embedding using GloVe(Pennington et al., 2014), a pre-trained word embedding to generate 200-dimension vectors. Then we implement and compare two sentence embedding models: sum of words (baseline) and LSTM. Then we map the two sentence embeddings (200d) into a low-dimension space (100d) and concatenate them as the input. We then feed the input into a neural network classifier. The structure of the network is just three tanh layers (200d). Finally, the output layer uses a softmax classifier. All the layers, including sentence embedding layer are trained jointly.

Preprocessing data The sentences are of variable length, while TensorFlow requires fixed input size. Thus we performed padding by adding zero vectors at the end of short sentences. We visualized the sentence length distributions in Figure 1. The distributions are similar for the pairs. We truncated the sentence length to 28 for questions 1 and 31 for questions 2. Only 1% of the sentences are longer than this size and thus the length was trimmed.

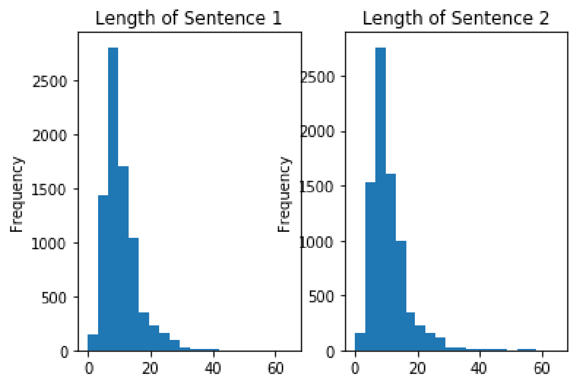

Figure 1: Distribution of sentence length: 8000 pairs from Quora dataset

3.2.1 LSTM encoder

Long Short Term Memory Network (LSTM) is first developed by Hochreiter and Schmidhuber (Hochreiter and Schmidhuber, 1997). Practically, a plain RNN has the problem of vanishing gradient and cannot capture long-dependencies when training over long sequences. However, Long Short Term Memory (LSTM) RNN addresses this problem by introducing a few gates which can control access to the cell state.

We briefly describe the idea of LSTM here (Olah, 2015). As illustrated in Figure 1, an LSTM memory cell consists of four layers:

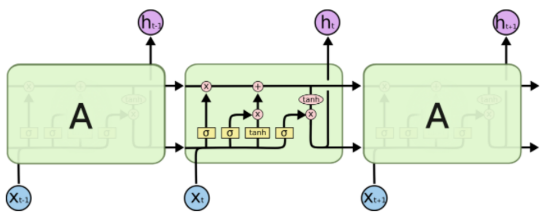

Figure 2: Illustration of LSTM

- Forget gate: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$, where input is current x_t and previous output h_{t-1}. The output (an quantity between 0 and 1 for each number in cell state C_{t-1}) decides the probability of what information to keep.

- This layers contains two components. An input gate: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$ decides which values to update. And a tanh layer: $\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$ which proposes a set of new candidates $\tilde{C}_t$

- Update cell state $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$, which basically means forgetting old information ($f_t * C_{t-1}$) and adding new information ($i_t * \tilde{C}_t$)

- Output gate: $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$ and $h_t = o_t * \tanh(C_t)$ which is a filtered cell state.

To implement LSTM encoders, the word embeddings will be fed into an LSTM model to generate sentence embedding.

4 Experiments

We evaluate our model on a subset of the Quora question pairs dataset available on Kaggle, which is released by Quora and contains 8,000 lines of potential duplicate question pairs. Each line contains six columns: ID of the question pair, IDs for each question in the pair, the full text for each question, and a binary indicators which marks whether the two questions in the pair are duplicate. We use accuracy, precision, recall and F1 score as our evaluation metrics. The dataset is split to be: train 75 %, development 15 %, test 15 %.

4.1 Feature-rich classifier model

We implemented sum of words embedding + random forest classifier as the baseline model. The performance over development data and test data are presented in Table 1 and 2. Using the same dataset which contains nearly 40k sentence pairs, the cross-unigram model performs the best among all in terms of F1-score, despite that they share almost the same score in terms of accuracy. We can safely draw the conclusion that the POS tagging knowledge helped improve the predicting of the logistic regression model used (logistic regression and random forest). For example, the premise ”How can I be a good geologist?” and the hypothesis ”What should I do to be a great geologist?”, the ending tokens of these two sentences share the same POS tags (can: MODAL, should: MODAL; I: PRON; be: VERB; a: DET; good/great: ADJ; geologist: NOUN/entity!) and are counted as having a collection of cross-unigram features (should, I, be, a, great, geologist). The intuition comes from the fact that verb-noun phrases and potential modifiers (adjectives, adverbs) tend to decide the meaning of a sentence than other word categories. Thus it is essential to extract the main structure from the sentence, here, word embedding is one way, the number of dimensions used to represent a word also matters, here we tried the vector representation of 50 dimension and 100 dimension, respectively. Also, the cross-bigram model helps raise the recall rate, in the way that bigram is also an important and useful feature in text understanding. Besides, as sentence inversion is pretty common, in order to recognize such inverse structures, cross-bigram-inverse model is also used to measure how well this feature can be used to predict the most.

Model	Acc.	Prec.	Recall	F1
Baseline	0.64	0.55	0.28	0.37
Cross-unigram	0.68	0.58	0.48	0.53
Cross-bigram	0.66	0.56	0.33	0.42
inverse-bigram	0.65	0.55	0.33	0.41

Table 1: Performance on development data

Model	Acc.	Prec.	Recall	F1
Baseline	0.66	0.62	0.33	0.43
Cross-unigram	0.69	0.59	0.47	0.52
Cross-bigram	0.69	0.65	0.35	0.46
inverse-bigram	0.69	0.64	0.36	0.46

Table 2: Performance on test data

4.2 LSTM sentence embedding model

All models are implemented in TensorFlow. We use cross-entropy as the training objective. The models are trained using Adam Optimizer with stochastic minibatch.

Tuning Hyper-parameters Choosing the right parameters is more of art than science. The methods for learning problem diagnosis and hyper-parameters tuning for our model refers to (Pompey, 2016). In this paper, the hyper-parameters we focus on are:

- Learning rate: The fixed learning rate we pass here is an initialization value. Adam optimizer will adjust the learning rate during the process of training.
- Dropout: dropout techniques are applied to all hidden layers of the neural network classifier and input and output of LSTM.
- L_2 regularization are applied to all models.

Results Finally, we select a learning rate of 0.001, a dropout keep rate of 0.5 for both sum of words model and LSTM model. The L_2 regularization strength is set to be 0.01 for sum of words and 0.1 for LSTM. The accuracy for development and test data are presented in Table 3. We didn't find that LSTM has an obvious advantage over the baseline sum of words. This is partly due to the over-fitting problem. Because of speed issue, we just trained on 8000 pairs of data, which makes the Bowman's data-intensive method too complex for this dataset.

Accuracy	Dev	Test
SumOfWords	0.63	0.63
LSTM	0.65	0.61

Table 3: Accuracy on dev and test dataset by different sentence model

To compare the sentence embedding performance for the models, we also investigated the sentence representation vectors for some examples. For duplicate sentence pairs shown in table 4, LSTM encoders always give higher similarity measure. Especially for the second pair, which has less syntatic similarity, sum of words embedding gives relatively low similarity, while LSTM still yield good result.

Sentence Pair	Sum	LSTM
How can I be a good geologist? What should I do to e a great geologist?	0.9205	**0.9901**
Astrology: I am a Capricorn Sun Cap moon and cap rising... what does that say about me? I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?	0.8972	**0.9884**

Table 4: Cosine Similarity for duplicate sentence pairs by two models

Qualitative Error Analysis Both models failed when the inference needs world knowledge and external resources. For example, for the duplicate sentence pair: *How is the new Harry Potter book 'Harry Potter and the Cursed Child'?* and *How bad is the new book by J.K Rowling?*, both models predict it as non-duplicate since the relationship between 'Harry Potter book' and 'J.K Rowling' cannot be inferred merely by the context. Thus, for the future work, we can consider incorporating external knowledge base and entity linking. They also both failed to correctly predict non-duplicate questions when the sentences largely share lexical and syntactic information. For example: *What can be done to reduce the pollution of India?* and *What is the best way to reduce pollution in india?*, are predicted as duplicate by both models. Most of the words are the same, and 'What can be done ' and 'What is the (best) way' are similar phrases. Attention model can potentially improve the prediction in this case, since it might will assign more attention weight to the word 'best'.

5 Conclusions and Future Work

In this paper, we implemented two models. One is feature-rich classifier model, which uses lexical and semantic information to build features such as cross-unigram, cross-bigram, inverse-bigram etc. We also implemented sentence embedding models based on sum of words and LSTM. All these methods outperform the baseline method.

Our evaluation is based on a subset of the original data. We are planning to evaluate our current methods on the complete dataset (a large corpus), since data-intensive models are more practically important.

Potential directions to improve current methods include but are not limited to:

- Implement batch normalization to alleviate LSTM over-fitting problem
- Improve the LSTM encoder by using bidirectional LSTM (biLSTM) to generate sentence embedding
- Implement lexical mapping and discover more promising features such as the mapping between acronyms and full names, named entity matching, etc.

References

Dasha Bogdanova, Cícero Nogueira dos Santos, Luciano Barbosa, and Bianca Zadrozny. 2015. Detecting semantically equivalent questions in online user forums. In *CoNLL*. volume 123, page 2015.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* .

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. *arXiv preprint arXiv:1508.02354* .

Nikhil Dandekar. 2017. Semantic Question Matching with Deep Learning. https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*. Citeseer, pages 45–52.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Matthew Honnibal. 2017. Deep text-pair classification with Quora's 2017 question dataset. https://explosion.ai/blog/quora-deep-text-pair-classification.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *EMNLP*. pages 891–896.

Christopher Olah. 2015. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Pascal Pompey. 2016. The art of deep learning (applied to nlp). https://cs224d.stanford.edu/reports/pascal.pdf.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*. volume 24, pages 801–809.