Duplicate Question Pair Detection with Deep Learning

Travis Addair
Department of Computer Science
Stanford University
taddair@stanford.edu

Abstract

Determining whether two questions are asking the same thing can be challenging, as word choice and sentence structure can vary significantly. Traditional natural language processing techniques such as shingling have been found to have limited success in separating related question from duplicate questions. Using a dataset of 400,000 labeled question pairs provided by question-and-answer forum Quora, we explore a series of deep learning methodologies for detecting duplicate question pairs: convolutional neural networks (CNNs), long short-term memory networks (LSTMs), and a hybrid model. All three models are built atop a siamese network architecture and multilayer perceptron concatenation for the final inference. Our empirical results show that LSTMs outperform CNNs in terms of accuracy, and that combining the two techniques provides no additional inference improvements. Moreover, all three deep learning techniques significantly outperform traditional NLP methods and simple multilayer perceptron baselines.

1 Introduction

Question-and-answer (Q&A) websites such as Quora[1] provide users with a platform to ask questions that other users on the site may answer. However, many of the questions being asked at any given time have already been asked by other users, usually with different wording or phrasing. Ideally, these duplicate questions would be merged together into a single canonical question, as doing so would provide a number of benefits:

- It saves the question asker time if their question has already been answered previously on the site. Instead of waiting minutes or hours for a response, they can get their answer immediately.
- Frequently repeated questions can frustrate highly engaged users whose feeds become polluted with redundant questions. Many users who answer questions in a particular topic see slight variations on the same question appearing many times in their feed, and this creates a negative user experience for them.
- Q&A knowledge bases have more value to users and researchers when there is a single canonical question and collections of answers, instead of having the knowledge fragmented and spread throughout the site. This reduces the time it takes for users to find the best answers, and allows researchers to better understand the relationship between questions and their answers.
- Having knowledge of alternative phrasings of the same question can improve search and discovery. Full text search is a valuable feature of Q&A sites, but its utility is

1 http://www.quora.com

limited by needing to query for near-exact question phrasing. Having multiple representation of the same question can improve this search process significantly for users.

We say that two questions are *duplicates* if the question askers expressed the same intent when writing the question. That is, any valid answer to one question is also a valid answer to the other. For example, the questions “What’s the fastest way to get from Los Angeles to New York?” and “How do I get from Los Angeles to New York in the least amount of time?” are said to be duplicates.

It’s worth noting that some questions have inherent ambiguity based on their texts, and we cannot say for certain that they express the same intent. For example, “How do I make $100k USD?” and “How do I get a hundred grand?” could be duplicates, if we assume that the “hundred grand” desired is in US dollars, but that may not be true. Often, any human labeling process will reflect this ambiguity, and introduce some amount of noise to the dataset.

In this paper, we propose a siamese neural network architecture for detecting duplicate question pairs, and apply three different encoding techniques to generate feature vectors for each question. These encoded features are concatenated into a single input vector that is then fed through a multilayer perceptron to infer the final label prediction. The encoding methods explored include a Convolutional Neural Network (CNN), a Long Short-Term Memory (LSTM) network, and a hybrid between the two that feeds LSTM output into the CNN.

2 Background

In traditional natural language processing (NLP), w-shingling (Broder [1]) has been successfully used to quantify the similarity between two text documents. However, duplicate questions can be rephrased in many ways, and so techniques such as this that rely on word overlap fall short for this task, as we show in our experiments.

At the task of sentence classification and sentiment analysis, CNNs have shown significant promise over traditional NLP techniques (Wu [2]). Taking sentence inputs truncated to a maximum length, the words of each sentence are converted into a matrix of pre-trained word embeddings derived using `word2vec` (Mikolov et al. [3]). The model has been shown to achieve good performance across a variety of sentence classification tasks including sentiment analysis. Bogdanova et al. [4] applied this technique to the problem of duplicate question pair detection using StackExchange[2] question data, and had very strong results on a very technical dataset (*AskUbuntu* forums).

With the release of Quora’s first public dataset[3], recent academic interest in duplicate question pair detect has been observed. Just prior to this paper’s publication, Wang et al. [5] applied bidirectional LSTMs to the problem of duplicate question pair detection and achieved state-of-the-art results when combined with hand-tuned cross-question features in a process they call “mutli-perspective matching”. This work formed the basis for attempting to apply an LSTM encoding to this task, and subsequently the creation of a hybrid LSTM with CNN encoding.

3 Approach

Figure1 illustrates the high level neural network architecture used in this study. We begin by passing each question into a separate tower of the network with identical parameters and weights, producing a “siamese network”. The raw questions, represented as single-dimensional vectors of vocabulary indexes, are then converted into pre-trained word

2 http://stackexchange.com/

3 https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

embeddings using GloVe (Pennington et al. [6]) in the embedding layer. The embedding matrix for each question is then passed through the encoding layer, that converts the word matrix into a single-dimensional feature vector. The feature vectors from each of the two independent networks are concatenated together using an form proposed by Bowman et al. [7]. Finally, the concatenated feature vectors are passed through a multilayer perceptron (MLP) that produces the final output.

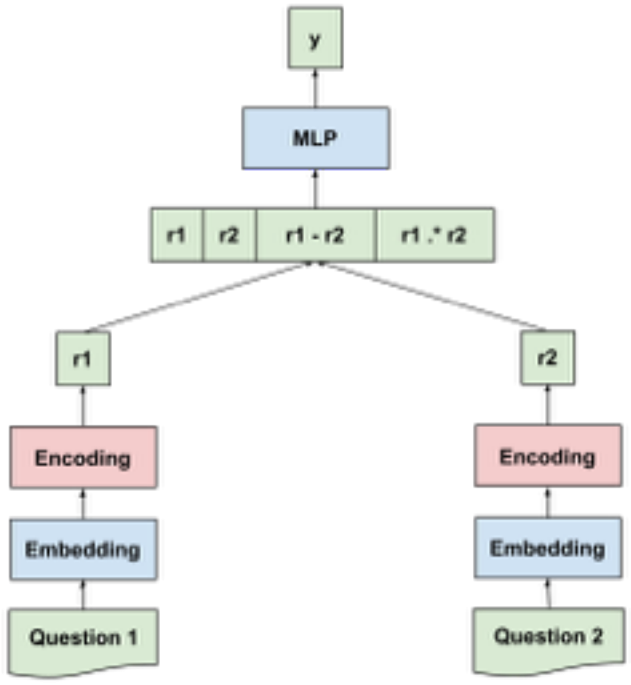

Figure 1: Siamese neural network architecture proposed

Three different encoding strategies are used in this study in the red encoding layer indicated in the figure above. The first encoding is a CNN similar to that proposed by Wu [2] and subsequently explored by Bogdanova et al. [4]. We use three filters sizes of (3 x d), (4 x d), and (5 x d), where *d* is the dimensionality of the word embeddings. This corresponds roughly to 3-gram, 4-gram, and 5-gram features over our sentences. Each of these filter sizes is applied 128 times to the embedding matrix. We use a ReLU nonlinearity over the filtered output, and apply max-pooling to the result. During training, dropout with a 0.5 keep probability is applied to the pooled output. Finally, all the pooled outputs are concatenated together, resulting in a final feature vector of size (1 x 328), i.e., 3 filter sizes * 128 filters per size.

In the second encoding, we apply a bidirectional LSTM similar to the one proposed by Wang et al. [5]. Each output state from the LSTM is encoded into a (1 x 5) vector. Since we perform both a forward pass and a backward pass over the word embedding matrix, this results in two (1 x 5) vectors per sentence position. Our sentences are constrained to be N words in length, which when concatenated tougher produces a (1 x 10N) feature vector, i.e., 5 element state * 2 directions * N words. We experimented with performing multiple layers over the LSTM, but found no improvement in performance, and so ultimately stuck with a single layer for the final results.

The third and final encoding strategy is a hybrid of the first two methods. First, a bidirectional LSTM is applied to the word embedding matrix, with output states of size (1 x d) where *d* is again the dimensionality of the word embedding. The output state for each direction of the LSTM are then concatenated together back into a (N x d) matrix. The two matrixes for each direction are then stacked together to form a single (N x d x 2) matrix. Next, the CNN used in the first encoding strategy is applied to this matrix, with the same ReLU activation, max-pooling, and dropout policies. This again produces a single (1 x 328) feature vector. We explored using multiple convolutions over the outputs, but found no gain in performance, and so stuck with a single convolution.

Common to all three encoding methods is the final multilayer perceptron that combines the siamese network feature vectors together. The input to this network taken from Bowman et al. [7], when introduced, provided a 2% performance gain over a simple concatenation of the two feature vectors. Specifically, we use as input:

$$(r1, r2, r1 - r2, r1 .* r2)$$

Where r1 and r2 are the output feature vectors from the first question and the second question, respectively. The third concatenated vector is the difference between the first and the second vectors (which will ultimately be symmetric given that each tower in the network uses the same model weights). Finally, we use as a third concatenation the element-wise product between the first and second vectors.

The single hidden layer of the MLP is constrained to have floor(sqrt(M)) nodes, where M is the length of the concatenated input vector described above. ReLU activation is again used for nonlinearity, and the final output layer consists of a single (1 x 2) vector, with the two elements corresponding to each of the two classes: non-duplicate and duplicate. We take the argmax over this vector to produce our final predication.

4 Experiments

The dataset used for this analysis was provided by Quora, released as their first public dataset as described above. It consists of 404352 question pairs in a tab-separated format:

- *id:* unique identifier for the question pair (unused)
- *qid1:* unique identifier for the first question (unused)
- *qid2:* unique identifier for the second question (unused)
- *question1:* full unicode text of the first question
- *question2:* full unicode text of the second question
- *is_duplicate:* label 1 if questions are duplicates, 0 otherwise

Of the over 400K question pairs, 149302 are duplicates, or roughly 37% of the full dataset.

A few notes on the sampling methodology used to generate the dataset. Quota has stated that the distribution of duplicates is not representative of questions across the site, which would result in a vanishingly small proportion of duplicates if questions were paired uniformly at random. Instead, negative samples are drawn from a pool of “related questions” that are similar in terms of word and phrase composition, but not classified as duplicates. Similarly, duplicate questions are drawn from the set of “merged questions” that have been human curated as duplicates. Because a single human user can choose to merge or not merge questions without moderator approval, there’s inherently some amount of noise in the labels.

In preprocessing the data for this task, we removed some 2413 question pairs of which 1157 were labeled duplicates. Specifically, we removed all question pairs whose combined word count across both questions was under 10 characters, as these questions tended to be nonsensical. We also removed any question pairs where at least one of the questions was over 59 words in length, as these tended to be long diatribes with very personal details, unlikely to be duplicated. This truncation allowed the model to be more efficient to train, but it would be worth investigating in the future ways of allowing these longer questions to be included in the dataset. For example, LSTMs can handle dynamic sequence lengths, and for CNNs we could consider simply truncating the questions that exceed our threshold.

Once the dataset was preprocessed, it was divided into a 90/10 split between training and test data. Of the 90% training dataset, 4% of that was withheld as a validation set during training. There were no hyper-parameters that were dynamically tuned at during training at runtime, but the validation set was used to check for overfitting. In assessing the performance of our models, we looked primarily at accuracy, but also reported on precision, recall, and F1 score. In order for our model to be considered non-trivial, we should expect an accuracy score above 63%, to account for the bias in the labels, otherwise our models could simply label all samples as non-duplicate and achieve this performance.

4.1 Baselines

We looked at two baseline models for the purpose of comparison in this study: shingling and a multilayer perceptron.

The first model is based off the work of Broder [1] using w-shingling to compare the similarity between the two questions using only their texts. In shingling, we represent each question as a bag of “shingles” or “n-grams” with *n* being from 1 to 4. The bags for each question are then compared using the Jaccard similarity coefficient:

$$J(S(q1), S(q2)) = S(q1) \cap S(q2) / S(q1) \cup S(q2)$$

Where S(q) is the set of shingles for the given question. This score is then compared against a threshold, with any coefficient above the threshold being declared a duplicate. Using binary search over the space of possible threshold values T in [0, 1], we obtained a best value for T of 0.13 on the training data.

Our second model used a simple feed forward neural network, a multilayer perceptron whose input is the same concatenated feature vector proposed by Bowman et al. [7] used in the other network architecture. Here, however, the inputs to the concatenation layer are two (1 x d) vectors — where d is the word embedding dimension — where each vector is the average of the word embeddings in the respective question. This produces a “bag word words” input vector. The concatenated vectors are then fed through a fully connected layer that outputs the confidence values for the two classes as the final output.

4.2 Results

In running our experiments, we used 64 sample batch sizes and trained for a total of 20 epochs. We chose N=59 as our sequence length for all models. Adam optimization was used with a learning rate of 0.001. Softmax cross entropy with logits was chosen for the loss. All of the implementation was written in TensorFlow, with separate run scripts for training and evaluation.

Across all three of the models studied, we found a similar pattern of convergence. The model accuracy would sharply increase over the first training epoch, then quickly flatline after about 2 to 3 epochs, as shown in Figure 2. We observe a similar pattern in the loss, where it decreases for the first few epochs, then begins to increase consistently as the magnitude of the weights increases, as shown in Figure 3.

Overall, we found that the three deep learning models significantly out-performed the baselines across all metrics, but none of the three deep learning models stood out as being noticeably more expressive than the others. The LSTM was notable for having slightly better overall performance than the CNN, but being significantly faster to train and perform inference.

Most surprising was that the hybrid model did not provide any gain in performance over the LSTM whatsoever, and in some ways may have actually underperformed it. It’s possible that the encoding of the LSTM captured some amount of information that was otherwise lost during CNN filtering process.

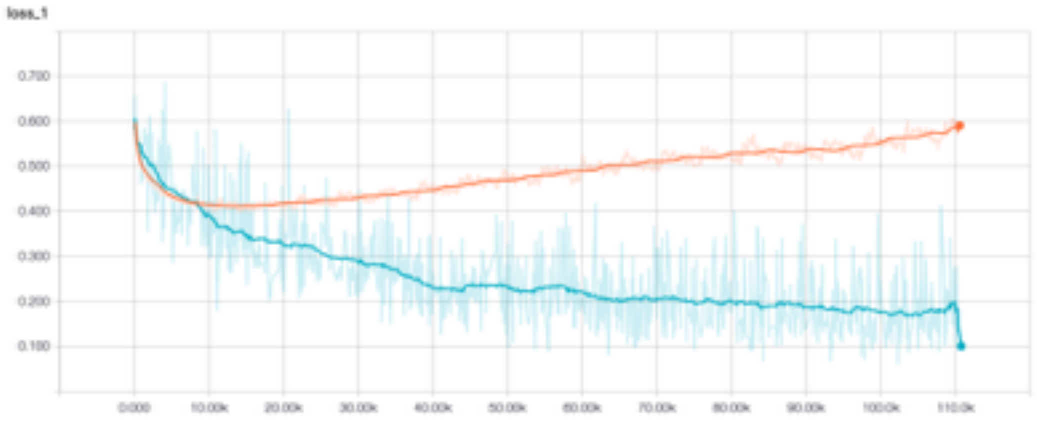

Figure 2: Accuracy of the best performing LSTM model over training steps

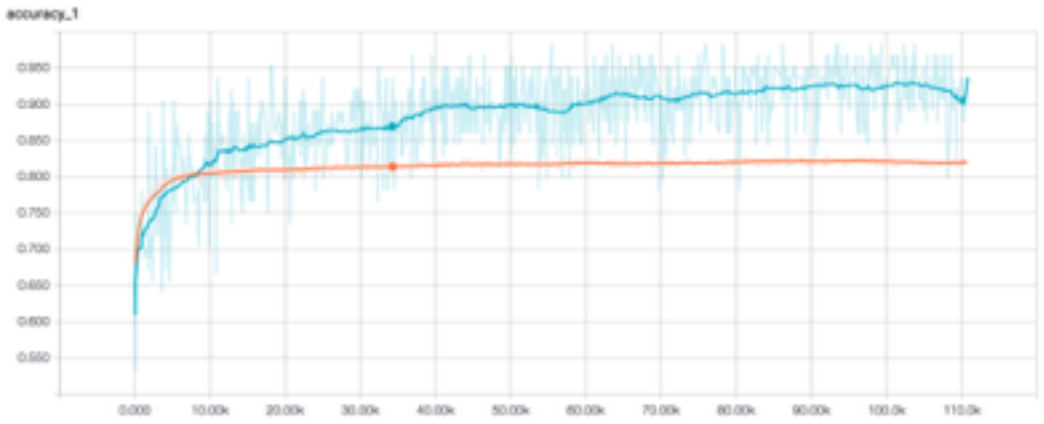

Figure 3: Loss of the best performing LSTM model over training steps

In Figure 5, we see the complete comparison between the five models studied. Clearly, the deep learning models are significantly better than the baselines. Shingling does better tan than the trivial model, but only manages to achieve approximately 67% accuracy. The baseline MLP is a significant step up at 73% accuracy, but still falls well short of the 80+% accuracy achieved by all the deep learning models. The LSTM performs the best on all metrics except for precision, but it should be noted that during training the models often oscillated up and down along the precision-recall tradeoff, and it's difficult to say that at any fixed point the scores reported are a complete picture of the overall performance.

Table 1: Model performance comparison

	Shingling	**MLP**	**CNN**	**LSTM**	**LSTM + CNN**
Accuracy	0.6657	0.7263	0.8027	**0.8107**	0.8105
Precision	0.5151	0.5878	**0.7102**	0.6862	0.7004
Recall	0.7297	0.7245	0.7349	**0.8441**	0.7994
F1	0.6039	0.6490	0.7223	**0.7570**	0.7466

In all of these models, we used a word embedding pre-trained using the GloVe algorithm using 27 billion Twitter tweets. The vectors chosen were 200d, as these were the largest dimensionality vectors available with this dataset. Using 300d vectors with Wikipedia data was also explored, but training time was prohibitively slow, and Twitter vectors appeared to outperform 200d Wikipedia vectors on this dataset.

5 Conclusions

Deep learning methods clearly outperform the baselines for the task of duplicate question pair detection. The siamese network architecture achieved a 7% performance gain over a simpler multilayer perceptron architectures, and a 14% gain over the shingling method. The LSTM method had comparable and slightly better performance than the CNN, but was much faster to train. A hybrid model combining both the LSTM and the CNN together produced no noticeable performance gain, but was much slower to train overall. Using the general siamese network architecture provided good results, and the multilayer perceptron combination layer had success at combining the two feature vectors together.

In terms of possible extensions to the model, we suspect that using word embeddings trained on Quora data could provide a boost in performance. Perhaps most promising, however, is the idea of moving away from a siamese network architecture towards the multi-perspective matching described by Wang et al. [5]. It would be interesting to find a way to achieve this combining of question features within a combined CNN layer. Overall, however, it's clear that deep learning is a powerful tool when applied to this problem that shows a lot of promise going forward.

All of the code for this study is freely available on GitHub:

https://github.com/tgaddair/quora-duplicate-question-detector

Acknowledgments

Special thanks to Kevin Clark for his mentorship during this project. The idea of the concatenation layer, using a CNN, and comparison to other studies on this dataset were provided by him.

References

[1] Broder, A. (1997) On the resemblance and containment of documents. *Proceedings of the Compression and Complexity of Sequences 1997,* SEQUENCES'97, Washington, DC, USA. IEEE Computer Society.

[2] Yoon Kim. (2014) Convolution neural networks for sentence classification. *Proceedings of the 2015 Conference on Empirical Methods for Natural Language Processing*, pages 1746-1751. Doha, Qatar.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. (2013). Efficient estimation of word representations in vector space. *Proceedings of International Conference on Learning Representations,* ICLR 2013, Scottsdale, AZ, USA.

[4] Dagna Bogdanova, C´ıcero dos Santos, Luciano Barbosa, and Bianca Zadrozny. (2015). Detecting

Semantically Equivalent Questions in Online User Forums. *Proceedings of the 19th Conference on Computational Language Learning*, pages 123–131, Beijing, China, July 30-31, 2015.

[5] Zhiguo Wang, Wael Hamza, Radu Florian. (2017). Bilateral Multi-Perspective Matching for Natural Language Sentences. https://arxiv.org/abs/1702.03814.

[6] Jeffrey Pennington, Richard Socher, Christopher D. Manning. (2014). GloVe: Global Vectors for Word Representation. *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532-1543. http://www.aclweb.org/anthology/D14-1162.

[7] Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, Christopher Potts. (2016). A Fast Unified Model for Parsing and Sentence Understanding. https://arxiv.org/pdf/1603.06021.pdf.