Detecting Duplicate Questions with Deep Learning

Yushi Homma
Department of Mathematics
Stanford University
yushi@stanford.edu

Stuart Sy
Department of Computer Science
Stanford University
stuartsy@stanford.edu

Christopher Yeh
Department of Computer Science
Stanford University
chrisyeh@stanford.edu

Abstract

In this paper, we explore methods of determining semantic equivalence between pairs of questions using a dataset released by Quora. Our deep learning approach to this problem uses a Siamese GRU neural network to encode each sentence, and we experiment with a variety of distance measures to predict equivalence based on the sentence vector outputs of the neural network. We find that while logistic regression on the pure distance measures produces decent results, feeding a concatenation of different transformations of the output sentence vectors through another set of neural network layers yields significantly improves performance to a level comparable to current state-of-the-art models. In addition, we demonstrate data augmentation techniques that can be used to improve Siamese neural network model performance.

1 Introduction

Detecting semantic similarity and equivalence in sentences has many applications in natural language understanding, ranging from paraphrase identification to evaluating machine translation [7]. In this project, we focus on detecting semantically equivalent (duplicate) questions, a prevalent problem in online Q&A forums like Stack Overflow and Quora, for which combining the answers for duplicate questions asked by their users improves the efficiency and the quality of their service. We use the definition of semantic equivalence employed by Bogdonova et al. [1]:

Def. Two questions are *semantically equivalent* if they can be answered by the exact same answers.

Recent developments in deep learning methods have made significant gains in tasks like semantic equivalence detection, surpassing traditional machine learning techniques that use hand-picked features. We build on this existing work by exploring different configurations of deep neural networks to identify duplicate pairs of questions. We perform numerous experiments using Quora's "Question Pairs" dataset,[1] which consists of 404,351 pairs of questions labeled as 'duplicates' or 'not duplicates'. We assume that questions marked as duplicates in the Quora dataset are semantically equivalent since Quora's duplicate question policy[2] concurs with our definition of semantic equivalence above.

[1] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
[2] https://www.quora.com/Whats-Quoras-policy-on-merging-questions

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

id	qid1	qid2	question1	question2	is_duplicate
447	895	896	What are natural numbers?	What is a least natural number?	0
1518	3037	3038	Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0
3272	6542	6543	How do you start a bakery?	How can one start a bakery business?	1
3362	6722	6723	Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1

Figure 1: Sample of the Quora "Question Pairs" dataset

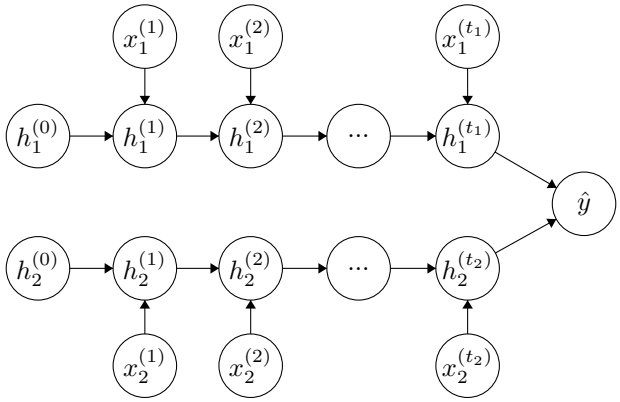

Figure 2: General architecture of Siamese neural network: x_1 and x_2 represent the inputs, h_1 and h_2 represent the hidden states, and $\hat{y}$ is the predicted similarity.

2 Related Works

Detecting semantically equivalent sentences or questions has been a long-standing problem in natural language processing and understanding. As Dey et al. [5] demonstrate, traditional machine learning algorithms such as Support Vectorm Machines (SVMs) using hand-picked features and extensively preprocessed data perform well on the SemEval-2015 dataset. They argue that the performance of deep learning methods is heavily limited by the small, noisy datasets that they are trained on.

Nonetheless, deep learning techniques have made considerable progress in recent years. Most deep learning methods for detecting semantic equivalence rely on a "Siamese" neural network architecture [3] that takes to two input sentences and encodes them individually using the same neural network. The resulting two output vectors are then compared using some distance metric, as shown in Figure 2. This approach is used successfully by both Bogdanova et al. [1] and Sanborn-Skryzalin [7]. Bogdanova et al. found that pairing a convolutional neural network (CNN) with a cosine-similarity distance measure was more effective than traditional methods of using Jaccard similarity or SVMs in identifying duplicate questions in a StackExchange dataset. Sanborn and Skryzalin [7] compared the use of recurrent neural networks (RNNs) and recursive neural networks with traditional machine learning methods and found that recurrent neural networks performed the best on the SemEval-2015 dataset.

To date, the only published results on the Quora dataset come from Wang et al. [8]. Observing that the encoding procedure in Siamese networks does not provide any interaction between the two input sequences, they instead propose a bilateral multi-perspective matching LSTM model. Their "matching aggregation" approach performs better than the Siamese CNNs and LSTMs that they tested. We, however, will be focusing on optimizing the use of Siamese neural network methods in this project for the task.

3 Approach

3.1 Data Preprocessing

As shown in Figure 1, the Quora dataset provides a completely labeled dataset of pairs of questions. In the preprocessing step, we first tokenize the sentences in the entire dataset using the Stanford Tokenizer[3] included in version 3.7.0 of the Stanford CoreNLP Suite. We chose this tokenizer because it is the same tokenizer used for GloVe word embeddings. All characters are converted to lowercase, and we disabled the `ptb3Escaping` option. We then replaced each token with its corresponding ID in the GloVe vocabulary. For those tokens not represented in the GloVe vectors ($< 0.9\%$ of the Quora dataset's tokens), we assigned the specified '`UNK`' ID.

The input questions vary significantly in length, from empty (0-length) up to to 237 words. In order to batch our computations during training and evaluation using matrix operations, we needed the input questions to all have a fixed length. To do so, we padded the shorter sentences at the beginning with a designated zero-padding ID and truncated the longer sentences to the same standardized length, which is a hyperparameter to our model.

Lastly, we split our data into training, development (validation), and test sets. In order to produce results that can be compared to those from Wang et al. [8], we use their same test set.[4] For evaluating our models and tuning their hyperparameters, we randomly split the remaining non-test data into a training set and a development set, containing 85% and 15% of the question pairs, respectively.

3.2 Sentence Encoding

In this project, we explore two types of neural networks to encode each sentence: the recurrent neural network (RNN) and the gated recurrent unit (GRU). During training, we update the word embeddings as well as the weights and biases within the RNN/GRU cells. Each of these cells contains a single layer with the $\tanh$ activation function. The weights and the biases in the encoding layer are shared for both questions in the pair so that the network is smaller and easier to train. Each of these neural networks outputs a sentence vector of dimension H, the hidden state size in the RNN/GRU cells.

We initialize the word embeddings to the 300-dimensional GloVe vectors pre-trained by Pennington et al. [6] on the Wikipedia 2014 and Gigaword 5 datasets. The weights for the cells are initialized using Xavier initialization while the biases are zero-initialized.

3.3 Distance Measure

Once the questions of each pair are individually encoded into sentence vectors, we need to combine them in some way to predict whether or not the pair is a duplicate. The first method we use involves calculating some distance between the sentence vectors and running logistic regression to make the prediction,

$$\hat{y} = \sigma(a \cdot d(\boldsymbol{h}_1, \boldsymbol{h}_2) + b), \tag{1}$$

where σ is the logistic or sigmoid function, $a, b \in \mathbb{R}$ are learned parameters, and $d : \mathbb{R}^{\mathbb{H}} \times \mathbb{R}^H \to \mathbb{R}$ is a distance function between the two sentence vectors.

Inspired by previous work from Bogdanova et al. [1], we initially tried cosine distance. We also tested Euclidean (L_2) distance and weighted Manhattan distance.

$$d_{\cos}(\boldsymbol{h}_1, \boldsymbol{h}_2) = \frac{\boldsymbol{h}_1 \cdot \boldsymbol{h}_2}{||\boldsymbol{h}_1||\,||\boldsymbol{h}_2||} \tag{2}$$

$$d_{\text{euc}}(\boldsymbol{h}_1, \boldsymbol{h}_2) = ||\boldsymbol{h}_1 - \boldsymbol{h}_2|| \tag{3}$$

$$d_{\text{w}-\text{man}}(\boldsymbol{h}_1, \boldsymbol{h}_2) = \boldsymbol{w} \cdot |\boldsymbol{h}_1 - \boldsymbol{h}_2| \tag{4}$$

where $\cdot$ is the dot-product, $||\cdot||$ is the L_2 norm operator, and $\boldsymbol{w} \in \mathbb{R}^H$ is a learned vector of weights.

However, even though we can test multiple distance measures, we do not know what a "natural" distance measure is in the sentence vector space encoded by the neural network. To address this

[3] https://nlp.stanford.edu/software/tokenizer.html

[4] Available at https://zhiguowang.github.io.

Table 1: Effect of regularization constant on accuracy and F1 score on the validation set for the model using a GRU with a 2-layer similarity network.

Regularization Constant	Acc. (Val)	F1 (Val)
0.01	0.8627	0.8105
0.001	0.8623	0.8081
0.0001	0.8631	0.8103

problem, we replaced the distance function with a neural network outputting a softmax over the two possible classes, leaving it up to this neural network to learn the correct distance function. We experimented using one layer and two layers in the neural network. In the 1-layer network, the prediction is

$$\hat{\boldsymbol{y}} = \text{softmax}(\text{ReLU}(\boldsymbol{v})\boldsymbol{U} + \boldsymbol{b}) \tag{5}$$

where $\boldsymbol{U} \in \mathbb{R}^{4H \times 2}$ and $\boldsymbol{b} \in \mathbb{R}^2$ are the learned weights and bias, and the input $\boldsymbol{v}$ to this neural network is a row concatenated vector:

$$\boldsymbol{v} = [\boldsymbol{h}_1 \quad \boldsymbol{h}_2 \quad (\boldsymbol{h}_1 - \boldsymbol{h}_2)^2 \quad \boldsymbol{h_1} \odot \boldsymbol{h_2}]. \tag{6}$$

We adapted this concatenated vector idea from a similar method used by Bowman-Gauthier [2] in their paper on sentence parsing. However, we realized that their concatenated vector used $h_1 - h_2$, which is not symmetric between h_1 and h_2, so we compared the results with using the squared difference $(h_1 - h_2)^2$. We found that the squared difference improved the model, so we replace $h_1 - h_2$ with $(h_1 - h_2)^2$ in our vector.

In the 2-layer network, we used the same concatenated vector $\boldsymbol{v}$ and feed it through an additional hidden state $\boldsymbol{v_1}$ with dimension H.

$$\boldsymbol{v_1} = \text{ReLU}(\text{ReLU}(\boldsymbol{v})\boldsymbol{U_1} + \boldsymbol{b_1}) \tag{7}$$

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{v_1}\boldsymbol{U_2} + \boldsymbol{b_2}) \tag{8}$$

3.4 Loss

For the distance-logistic regression method, we use the mean-squared error as our loss function. For the two-layer neural network softmax method, we use cross-entropy loss. To avoid overfitting, we use L_2 regularization for all our weights and biases and found that a regularization constant of $\lambda = 0.0001$ achieved the highest accuracy on our validation set (Table 3.4). For the distance-logistic regression method, the total loss is

$$Loss_{dist} = \frac{1}{N}||\hat{\boldsymbol{y}} - \boldsymbol{y}||^2 + \lambda||\boldsymbol{\theta}||^2, \tag{9}$$

and for the softmax method, we have the total loss of

$$Loss_{soft} = CE(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda||\boldsymbol{\theta}||^2 \tag{10}$$

where CE denotes the cross-entropy loss function. The vectors $\hat{\boldsymbol{y}} \in (0,1)^N$ and $\boldsymbol{y} \in \{0,1\}^N$ are the predictions and ground truth labels for all N training examples, and $\boldsymbol{\theta}$ is a 1-dimensional vector containing all of the weights and biases used in the model. To optimize the loss, we use the Adam optimizer and backpropagation algorithm provided by Google's TensorFlow library.

4 Experiments

4.1 Model Types

As mentioned in Section 3, we experimented using RNNs and GRUs with different distance metrics to make our model. Table 2 shows the accuracy and the F1 scores (on the validation set) of the different model types we tested.

Since the outputs $\hat{\boldsymbol{y}}$ of our models are real numbers between 0 and 1, we round to the nearest integer to make a final prediction before computing accuracy and F1 scores. We define accuracy as

$$\text{Acc} = \frac{\text{Number of correct predictions}}{\text{Number of data examples}} \tag{11}$$

Table 2: Comparison of the best accuracy and corresponding F1 score of the models we experimented with on the validation set.

Model	Acc. (Val)	F1 (Val)
RNN, cosine distance	0.7737	0.7007
RNN, L_2 distance	0.7860	0.7077
RNN, weighted Manhattan distance	0.7991	0.7160
RNN, 1-layer similarity network	0.8061	0.7320
RNN, 2-layer similarity network	0.8094	0.7220
GRU, cosine distance	0.7708	0.7170
GRU, L_2 distance	0.8309	0.7735
GRU, weighted Manhattan distance	0.8422	0.7845
GRU, 1-layer similarity network	0.8633	0.8132
GRU, 2-layer similarity network	0.8631	0.8103
GRU, 1-layer similarity network, augmented data	0.8668	**0.8191**
GRU, 2-layer similarity network, augmented data	**0.8680**	0.8173

We found that our best model is the GRU that uses a 2-layer similarity network to predict duplicity from the hidden states. We can also see that among the distance methods, the weighted Manhattan distance performed the best, followed by L_2 distance and cosine distance. This shows that the way the algorithm separates sentences in the sentence vector space is not simply by cosine distance or Euclidean distance.

4.2 Spellcheck Preprocessing

Upon noticing that there were misspelled words in some of the questions in the dataset, we tried using a spell-check on the data during preprocessing to improve our results. We used the `autocorrect` Python package to correct the spelling of words longer than 3 letters to their closest correct spelling. However, this actually decreased performance compared to the uncorrected data. We believe there are two explanations for this. First, since the GloVe vectors were trained on unsanitized datasets including Wikipedia, the GloVe vocabulary already includes many common spelling mistakes, such as the the misspelled contraction "theyre." As a result, over 99.1% of the words in the Quora dataset are found in the GloVe vocabulary, so the room for improvement from additional spell-checking is limited. In addition, some words not in the GloVe vocabulary that would have been marked "UNK" were incorrectly changed by the spellcheck mechanism to real words that had no relation to the original word.

4.3 Data Augmentation

We noticed that our training accuracy and F1 scores were exceeding 99% and 0.99, respectively, far greater than the results on the validation set. However, these numbers changed very little as we varied the regularization constant λ (Table 3.4). Thus, we looked toward data augmentation as another approach to reduce overfitting.

To create artificial negative examples (non-duplicate pairs) to the existing dataset, we assumed that two questions randomly chosen from two different rows in the dataset are non-duplicate questions. To create more relevant artificial samples, we only chose new pairs of questions with a Jaccard similarity greater than 0.1.

We also added an equal number of artificial positive samples (duplicate pairs) to offset the increase in negative examples. We accomplished this in two ways. 50% of the new duplicate pairs come from pairing random questions to themselves. The other 50% comes from reversing the order of existing duplicate pairs. Note that reversing the random order of existing duplicate pairs would typically be less useful in a traditional Siamese network which is fully symmetric. However, our concatenated vector $\boldsymbol{v}$ is not completely symmetric, which enables this data augmentation technique to work well.

In total, we doubled the size of our training set, split evenly between duplicate and non-duplicate pairs. This resulted in an increase in accuracy of approximately 0.5% on the validation set for our

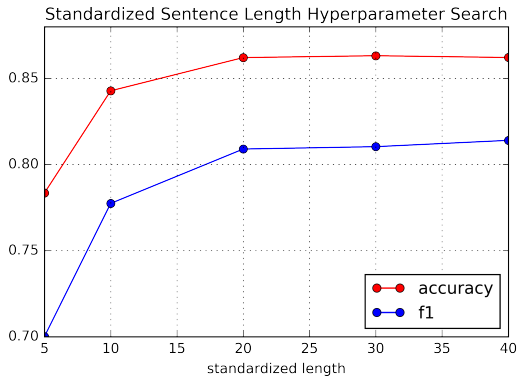

Figure 3: Graph of standardized sentence length vs. accuracy (blue) and F1 (red) for model using GRU with 2-layer similarity network

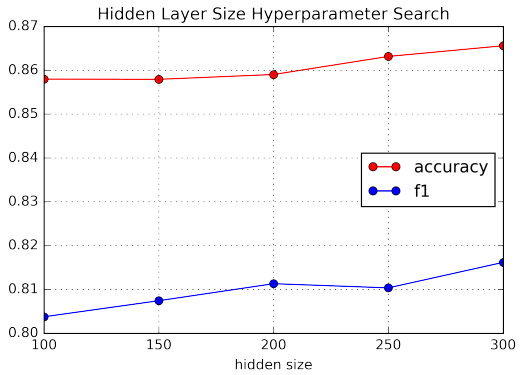

Figure 4: Graph of hidden layer size vs. accuracy (blue) and F1 (red) for model using GRU with a 2-layer similarity network

best model, the GRU using a 2-layer similarity network. With this experiment, we showed that these data augmentation techniques can be a valid and valuable way to improve Siamese models in general. Given more time to fine-tune this technique, we believe that data augmentation has tremendous potential in improving the results even more.

4.4 Hyperparameter Search

To optimize our model's performance, we conducted a search over its hyperparameter space. In particular, we focused on tuning the size of the neural network hidden layer and the standardized length of our input sentences.

During the tuning of the standardized sentence length hyperparameter, we found that increasing the standardized length generally led to an increase in accuracy and F1 scores, as more information could be incorporated from longer questions, while shorter questions did not suffer from additional padding. However, our accuracy scores began to plateau at a standardized length of 30. This makes sense since only 1% of the questions in the dataset have over 30 words. A table and graph of our results from tuning the standardized length of our input sentences can be found in Figure 3 respectively.

Similar to the standardized sentence length parameter, we found that increasing the size of our hidden layer sentence representation generally led to an increase in model performance, as shown in Figure 4. While a hidden layer size of 300 showed slightly better accuracy, we settled on 250 as a good balance between accuracy and training time.

Table 3: Comparison of Tuned Models on Test Set

Model	Acc. (Test)	F1 (Test)
GRU, weighted Manhattan distance	0.8235	0.8161
GRU, 1-layer similarity network	0.8447	0.8391
GRU, 2-layer similarity network	0.8431	0.8356
GRU, 1-layer similarity network, augmented data	0.8524	0.8474
GRU, 2-layer similarity network, augmented data	0.8495	0.8418
BiMPM*	0.8817	–

4.5 Results and Discussion

After tuning the hyperparameters (standardized length and hidden layer size) on the validation set, we ran our models on the test sets. The results are shown in Table 3, where we include Wang et al.'s BiMPM model for comparison.

Our best model, the Siamese GRU using a 2-layer similarity network trained on the augmented dataset, achieved 85.0% accuracy on the test set split, comparable to the 88.17% accuracy reported by Wang et al. [8]. It is likely that the "matching aggregation" framework used by Wang et al. allowed them to capture more semantic information from the input sentences by using interactions between them as they were encoded. However, our addition of the vector of concatenated hidden state transformations fed through additional neural network layers and use of data augmentation allowed us to perform much better than their implementation of a Siamese LSTM neural network model (79.6% accuracy).

5 Conclusion

Our results build on previous research on using Siamese networks for identifying semantically equivalent questions. In particular, we provide two novel contributions.

First, passing the concatenated vector $\boldsymbol{v}$ of hidden state transformations through a neural network is much better at determining semantic equivalence than pure distance measures. Previous work by Bowman et al. [2] only used cosine distance, which we show is the least promising distance measure.

Second, a GRU neural network is able to extract the majority of the semantic meaning from a sentence from the first 10 to 15 words. This result is somewhat surprising given Dey et al.'s [5] claim that deep neural networks are unsuitable for learning semantic meaning on short-text content.

Lastly, we demonstrate that data augmentation can be more effective than L_2 regularization for reducing overfitting, especially in asymmetric Siamese neural nets where there are natural data augmentation techniques. While we did not have enough time to perfect this technique, we believe it holds significant potential.

For future work, we see several possible extensions of our model. Firstly, other researchers have demonstrated that weighting the word embeddings by their *TF-IDF* scores can improve performance on semantic textual similarity tasks [4]. The basic idea is to give rare words greater weight. In addition there is room for improvement by using "matching aggregation" techniques, as proposed by Wang et al. [8], that allow interactions between the input sentences as they are encoded through the neural networks.

6 Acknowledgments

We acknowledge the contribution of the "Question Pair Dataset" by Quora and the GPU computing resources of Microsoft Azure. We also acknowledge the contribution of the GloVe word vectors trained on the Wikipedia and Gigaword dataset by Jeffrey Pennington, Christopher Manning, and Richard Socher. We thank the Stanford NLP Group for their contribution of the Stanford Tokenizer. We thank Christopher Manning, Richard Socher and CS 224N Teaching Staff for guidance and the assignment 3 RNN/GRU general architecture that we modified for the purposes of our project. We

would especially like to thank Kevin Clark for his mentorship and guidance throughout the project, giving us helpful comments and pointing us to relevant research.

References

[1] Dasha Bogdanova, Cicero dos Santos, Luciano Barbosa, and Bianca Zadrozny. Detecting semantically equivalent questions in online user forums. *Proceedings of the 19th Conference on Computational Natural Language Learning*, 1:123–131, 2015.

[2] Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Chrisopher D. Manning, and Christopher P. Potts. A fast united model for parsing and understanding. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.*, 1, 2016.

[3] Jane Bromley, James W. Bentz, Leon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Sackinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:669–688, 1993.

[4] Tomáš Brychcın and Lukáš Svoboda. Uwb at semeval-2016 task 1: Semantic textual similarity using lexical, syntactic, and semantic information. *Proceedings of SemEval*, pages 588–594, 2016.

[5] Kuntal Dey, Ritvik Shrivastava, and Saroj Kaushik. A paraphrase and semantic similarity detection system for user generated short-text content on microblogs. In *COLING*, volume 16, pages 2880–2890, 2016.

[6] Richard Socher Jeffrey Pennington and Christopher D. Manning. Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[7] Adrian Sanborn and Jacek Skryzalin. Deep learning for semantic similarity. 2015.

[8] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. 2017.