Semantic Equivalence on Quora's 400,000 Question Pairs

Rohan Gupta (rmgupta2), Triveni Putti (tputti2), Ved Prakash (vpu2)

May 2, 2018

Introduction

In this paper, we tackle the task of creating different models to classify semantic equivalence on question pairs. We utilize Quora's 400,000 question pairs as released as part of a Kaggle competition. We utilize the models that achieved state-of-the-art performance in this task, but also compare it against a universal sentence encoder that was recently released by Google. Our paper finds that Google's sentence encoder was outperformed by a Siamese LSTM with Word2Vec embeddings.

Motivation

The main purpose of our project is to try to design a model that can do binary classification on two sentences and determine if they are semantically equivalent. While this task may seem trivial, it is actually fairly complex. There is a similar task of semantic similarity, which is widely explored. However, we think semantic equivalency is an independent task that can enable a lot of benefit in the field. Primarily, the classifier can allow for better paraphrasers which can enhance chatbots. In addition, semantic equivalence can be utilized to enhance search.

One of the primary issues with semantic equivalency is that it is inherently subjective. The boundaries at which a permuted sentence deviates from its original meaning is impossible to objectively derive. Theoretically, any sentence that is permuted in the slightest has different sentiment and linguistic style, therefore it is not semantically equivalent. However, most people would agree that semantic equivalency can *generally* be preserved while changing the sentence structure and words of a sentence. Therefore, the boundaries of semantic equivalency must be based on human consensus.

This presents a different problem, because such a classifier would need a labeled monolingual parallel corpus in order to create. Such a dataset would need to be based on large consensus in order to be generalized.

We also had another motivation. While working on this project, Google Brain released a universal sentence encoder [1]. We wanted to assess the proficiency of the encoder, and one of the best ways that we could do that is through semantic equivalency, and compare it against the state-of-the-art in the field.

Task

Our task is to implement several different models that can classify sentences with a binary output on whether a pair of sentences are semantically equivalent. We will then compare the proficiency of each of the models in order to derive insights on how they function, why a particular model is more suited for the task, and offer further suggestions to improve them.

Dataset

We are utilizing Quora's 400,000 parallel question pairs as our dataset [2]. The dataset consists of 400,000 pairs of questions, and labels as to whether each pair is semantically equivalent. In our experiments, we excluded pairs with non-ASCII characters. As a part of pre-processing, we performed tasks like tokenization and removal of common stopwords before applying the models. We also did some visualizations to understand the dataset better. The dataset was first introduced as part of a Kaggle competition. This dataset is large enough for a model to actually learn semantic equivalence, however, we are skeptical if the model would generalize. First, it would be limited to the domain of questions. In addition to that, the dataset's quality is in question. A lot of the labeling was generated through mechanical turk, which while is usually effective, may introduce bias into the classifier. As stated above, the unique nature of establishing semantic bounds requires that the dataset be built on consensus. However, we believe that Quora's dataset is sufficient to assess which models would be best suited for this task. For these models to be applied in a more generalized setting, a larger, higher quality monolingual parallel corpus will be needed.

Model 1 (Word2Vec + MaLSTM)

Our first model utilizes a Manhatten LSTM, or commonly referred to as a Siamese LSTM[3] . The Siamese LSTM is one where the LSTM layer is learned concurrently (and shared) for both questions at the same time and learned to reduce the cross entropy error. This architecture is commonly used in semantic similarity, and is perfect when assessing the difference between two inputs. It makes sense, because we don't want to have different outputs if question1 is switched with question2. The output of both of the inputs fed through the LSTM are then compared in a similarity function. The input of the questions are encoded through an embedding model. We utilized word to vector for this model architecture. The architecture of the model can be seen below[3]:

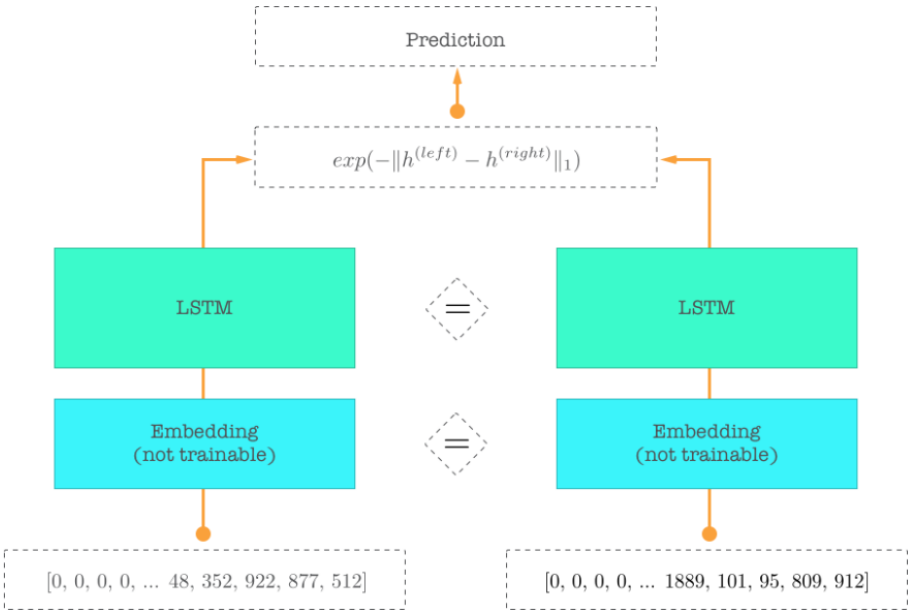

Figure 1 MaLSTM's architecture—Similar color means the weights are shared between the same-colored elements

The hyperparameters for this model are the Adadelta optimizer, 25 epocs, 50 hidden layers, a batch size of 64, and a gradient clipping norm of 1.25.

Model 2 (Google Encoder + DNN)

In this model we utilize Google's universal sentence encoder [1]. Google's sentence encoder can take a variable length sentence and convert it to a 512 dimensional vector. It is essentially a sentence-level version of embedding models and has been trained on a variety of data sources and a variety of tasks. This encoder so theoretically makes this task trivial. We first calculated a cosine similarity and visualized the distribution of this similarity score for the question pairs that were duplicate (labeled as 1) vs. the ones that were not (labeled as 0). Fig. 2. shows the distribution, where based on the similarity score, we found it difficult to distinguish between the two classes. Hence, we chose to implement a Deep Neural Network (DNN).

A simple DNN fed with the encoding of the question pairs should be able to produce effective results. We tested our model with two sets of features. One has 1024 features, which are simply the embeddings of each of the questions. The second had 1536 features and consisted of the two embeddings as well as the position difference between the two embeddings. The second model learned quicker, but in the end only achieved a miniscule increase in proficiency. We also tried the 1536 feature model with two different activation functions: Tanh and ReLu. ReLu performed better than Tanh. Further, a deeper neural network was also tried with three hidden layers of 30 units each and was run upto 100 epochs. The accuracy did not improve significantly.

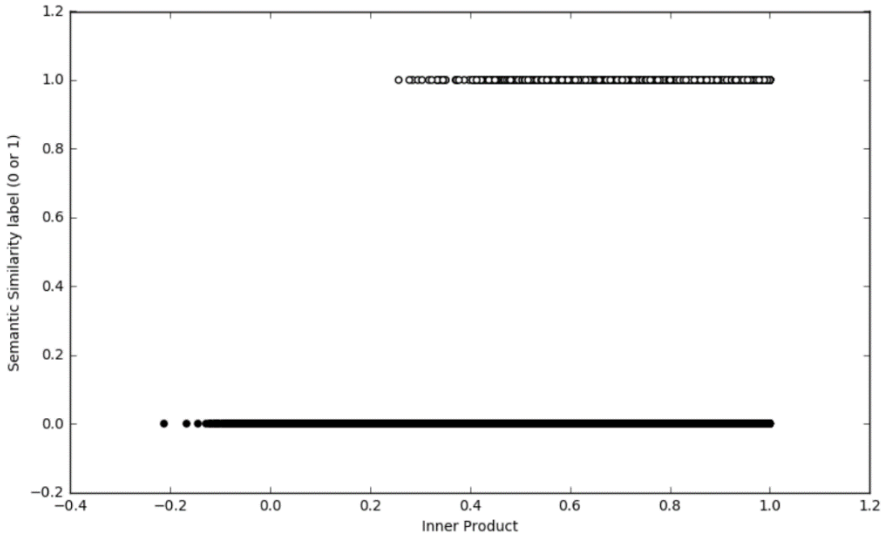

Figure 2: Distribution of similarity score for the two classes (labeled 0 and 1)

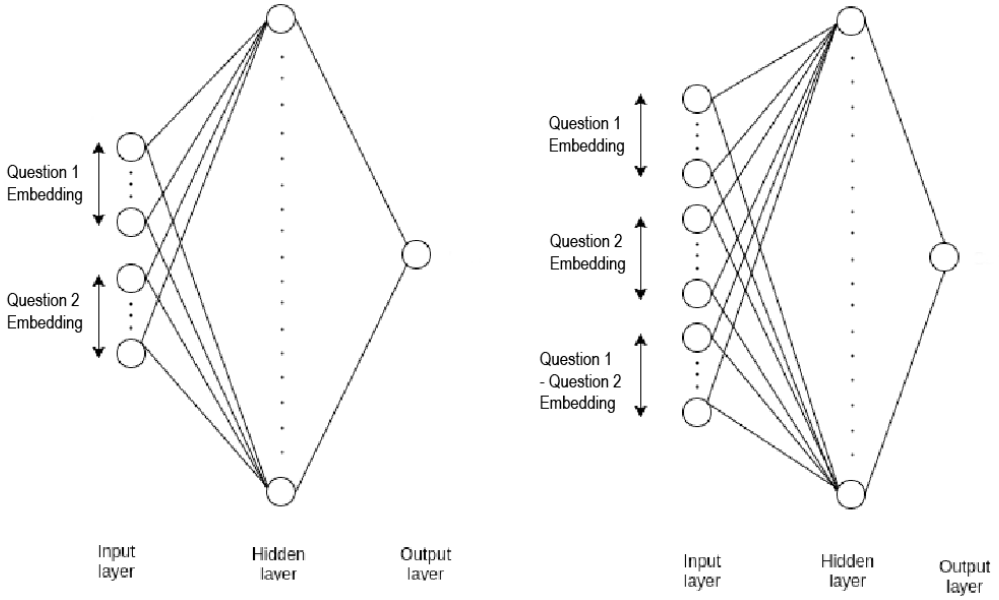

Figure 3: DNN architecture with sentence embeddings

The final hyperparameters we chose were one hidden layer, with 8 hidden units, a ReLu activation function, 70 epochs, and a batch size of 10. We did not want to add more hidden layers since it would unnecessarily complicate the model without giving any better results.

Model 3 (GloVe + MaLSTM + A)

Our third model utilizes the same general architecture as the first model[4]. However, there are two different key differences. First, we utilize an attention mechanism that will compare key relational words between the two questions. Second, we utilize a GloVe embedding

model in order to improve the proficiency of the embeddings themselves. We believe this is just an enhanced version of our first model.
The hyperparameters we used here was 25 epoch, batch size 516 and a dropout equal to 0.2.

Experiment

We conducted many experiments, however, due to the amount of models that we trained, we were unable to fully parameter tune all three models. However, we did light tuning on parameters like epocs, type of activation function and number of hidden units, and we were able to achieve close to state-of-the-art. The dataset was split into 90% training data and 10% validation data. The following graphs show the training and validation accuracy for the three models implemented. We are reporting the validation accuracy because the test set given in the kaggle website does not contain labels in order for us to compare with and calculate test accuracy.

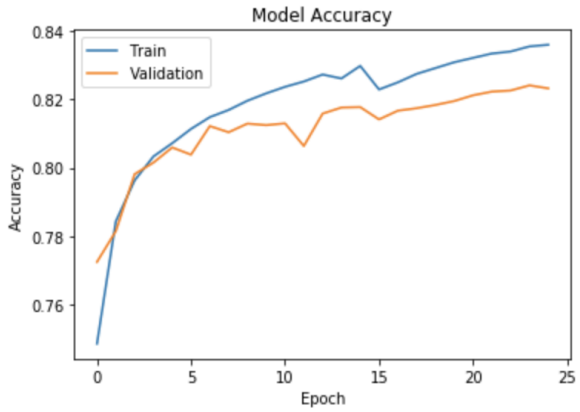

Figure 4: Word2Vec + MaLSTM accuracy

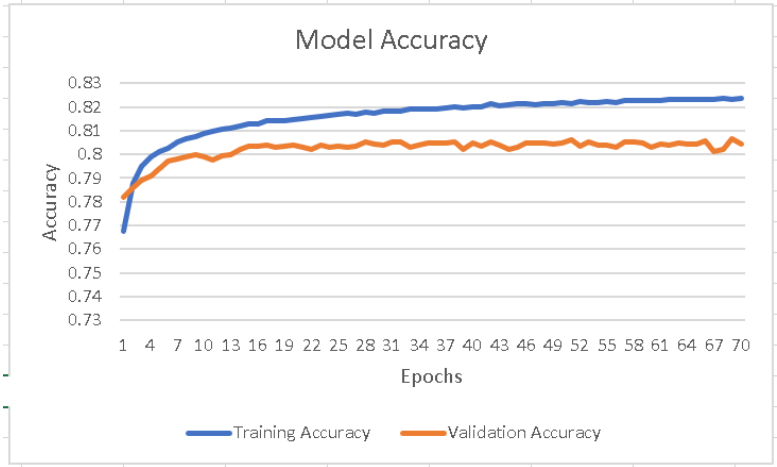

Figure 5: Google's Universal Sentence Encoder + DNN

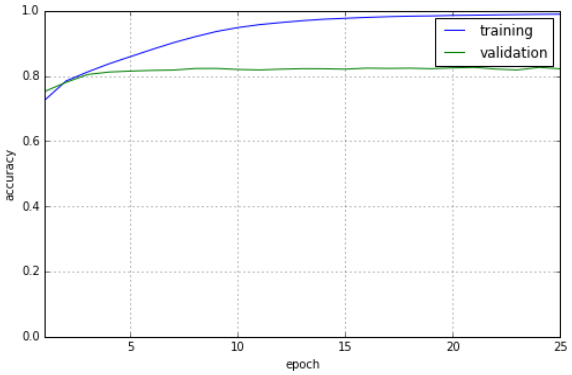

Figure 6: GloVe + MaLSTM + Attention accuracy

Model	**Training Accuracy**	**Validation Accuracy**	**Training Loss**	**Validation Loss**
Word2Vec + MaLSTM	83.60%	82.32%	12.10%	12.90%
Universal Sentence Encoder + DNN	82.33%	80.65%	37.46%	40.93%
GloVe + MaLSTM + Attention	98.96%	82.20%	2.91%	87.77%

Figure 7: Test results comparing all three models

There are many interesting things to note about these models. First, the MaLSTM with Attention had very high validation loss, and a very high training accuracy. This is a classical sign that this model is overfitting to the training data. Second, the simple Word2Vec with a MaLSTM was able to achieve the highest validation accuracy, indicating it is the state-of-the-art model (although it is not parameter tuned).

Conclusions/Further Work

Our experiments have found that the MaLSTM architectures, particularly the model with attention and GloVe have outperformed Google's sentence encoder. There are many insights that we derived.

First, obviously, the quality of the embedding models will directly impact the quality of the model. We propose utilizing sense to vector instead of word2vec or GloVe in order to achieve better performance. Because we have no control over Google's sentence encoder, it is hard to experiment on that. However, the architecture for the sentence encoder is based on a hierarchical skip-thought.

We predict that an english encoder for a zero-shot translation system will achieve far better results. However, it's important to note that implementation of Google's sentence encoder with a DNN is essentially trivial with current frameworks. The encoder is also a lot more

lightweight than traditional LSTMs. We predict that while Google's sentence encoder did not outperform current state-of-the-art in semantic equivalence, it most likely will when the encoder itself is improved.

References

[1] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, Ray Kurzweil et al. (2018) Universal Sentence Encoder, arXiv

[2] Quora. (2018) *Quora Question Pairs* [Question pairs and labels] retrieved from: https://www.kaggle.com/c/quora-question-pairs

[3] Cohen, E. (2017, June 7). How to predict Quora Question Pairs using Siamese Manhattan LSTM.

[4] Mueller, J., & Thyagarajan, A. (2016, February). Siamese Recurrent Architectures for Learning Sentence Similarity. In AAAI (pp. 2786-2792).