Semantic Similarity and Response Prediction for Programming Q&A

Rahul.Kulkarni@berkeley.edu Ryan.Rosich@berkeley.edu

W266 Spring 2020, UC Berkeley School of Information

Submitted 20 April 2020

Abstract

We are motivated to understand the challenges of applying generalized natural language models to a technical language domain such as computer programming Q&A in order to reduce the amount of manual effort required by a programmer to find satisfactory answers to their questions. We apply and analyze existing approaches of learning sentence similarities and response prediction for Q&A tasks to other domains. Our methods train supervised neural models to predict question similarity and best response. In our experiments, we augment our models with Universal Sentence Encoder and BERT models and observe strong results for question pair similarity prediction, but more modest results for response prediction.

Keywords: NLP, Deep Learning, Semantic Similarity, Response Prediction, Coding Q and A

Introduction

Programmers ask technical questions for coding tasks on various online forums, such as Stack Overflow. One such question might be “How to save a Python API result into a JSON file?”. Other users may ask related questions such as “How do I write JSON data to a file?” or “How to persist my JSON output?”. The questions may subsequently be annotated manually in online forums by users to indicate the question has previously been asked and resolved.

The asker of the question may also pick the best answer based on a set of responses. Some of the characteristics of such “best answers” may be that they satisfy the asker, are succinctly described, include technically accurate example code, or reference an authoritative document or wiki. Responses are also accompanied by an upvote count from the user community to indicate usefulness.

We are motivated to apply NLP techniques to augment the manual tasks of annotating duplicates and best responses by predicting semantically similar questions and the best answer from a collection of responses in order to reduce the amount of manual effort required by a programmer to find a satisfactory answer to their questions.

Deep learning models for Semantic Similarity and Response Prediction have shown increasingly good accuracy in recent years. We explore existing NLP research to apply such techniques to a different class of problems. NLI research on large corpuses for sentence entailment and contradiction provides an excellent basis for sentence pair classification (Bowman et al., 2017)[1]. Pretrained Sentence Encoding models such as Universal Sentence Encoder (USE) (Cer et al., 2018)[2] and

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018)[3] for transfer learning, provide state of the art performance for downstream tasks. Approaches for predicting the best response for a question from a collection of responses in a natural language conversation have shown good accuracy (Henderson et al., 2017; Yang et al., 2017).[4,5] We are motivated to leverage the techniques cited in the above literature for the work described in this paper.

Background

In recent years, Conversational Q & A systems have been performing exceptionally well. We are motivated by work on human-like chatbots (Adiwardana et al 2020.,)[6] and would like to explore a human like chatbot experience for specialized technical domains and complex tasks as demonstrated by the work in Iris (Fast et al., 2017)[7]. The Iris work applies NLP techniques to perform complex data-science tasks. As illustrated in (Ma et al., 2019)[8] applying state of the art models such as BERT for a new domain deteriorates the performance of deep neural models substantially and they propose a domain adaptation model for BERT to address the shortcomings. We would like to explore the domain-adaptation framework in the context of our work. Complex Q&A systems from open source knowledge sources require operating over a broader context for comprehension by combining information from multiple documents and facts to infer the answer. Open-domain Q&A research such as shown in (Dehghani et., al 2019)[9] using transformer architecture (Vaswani et al., 2017)[10] achieves results that meet state-of-the-art. The background research motivates us to explore in the direction of Q&A systems for complex tasks. In this paper we specifically address 2 tasks of question similarity and response prediction that will serve as building blocks for the broader research required.

Question and Answer Data Set

Our model relies on structured labelled data from Stack Overflow (Stack Overflow GCP.,)[11] for training. Our data is collected from posts available as a public dataset queried using Google BigQuery. We further narrow down the dataset to only include posts with the “Python” tag to generate our training and validation sets. Only questions with an accepted answer are included for the question and answer pairs. We limit the size of our data set in order to achieve optimal model performance given reasonable limitations on available computing resources. The text is tokenized and normalized, removing all stop words, punctuation, and code fragments from the question and answer text bodies. Our final dataset consists of 140K question pairs labelled as duplicates/non-duplicates and 100K question and answer pairs, with the accepted answer labelled. For the question similarity task, questions are paired together with a label indicating whether the questions have been marked as duplicates. In the best response task, a question and an answer are paired with the label representing an answer being marked as the “acceptable answer” for the question.

The two tables *Table 1* and *Table-2* show a sample of how our training dataset is constructed

Question 1	**Question 2**	**Duplicate?**
read 5th line python csv module	Best way to access the Nth line of csv file	Yes
assigning name lines read file	replace a part of file with another file	No

Table 1. Example question pairs and duplicate flag from the stackoverflow dataset

Question	Answer	Accepted Answer?
Understanding modular arithmetic in encryption example	The mod operator can be used in many different ways. In this case, it is used to constrain certain values and to make them "rotate"...	No
	The modular of 127 us to ensure you stay in ASCII bounds (basically so it's readable)...	No
	in the parts \n<code>...	Yes

Table 2: Example question-answer pairs and accepted answers flag

Methods

Model Architecture

We refer to our two tasks as *Task-1: Predict Semantic Similarity between question pairs* and *Task-2: Best Response for a question and answer pairs* in the rest of the paper.

Model 1: Two Tower Dot Product using USE

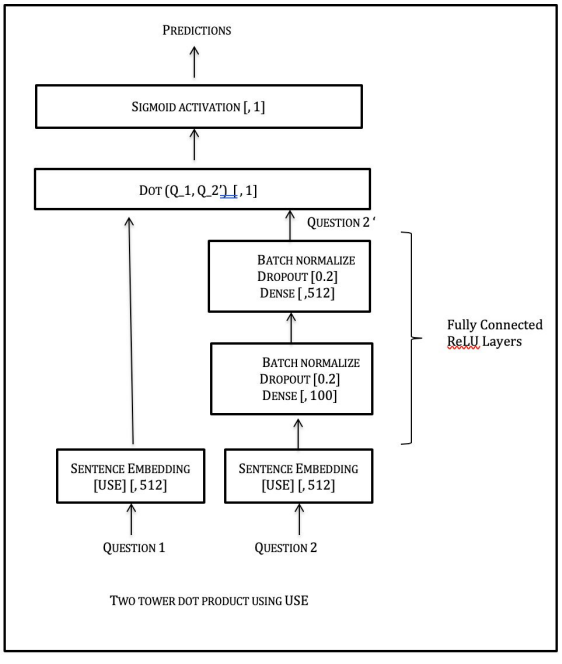


Fig.2 Two tower dot product model using USE for Task-1

We experimented with 4 model architectures described in this section for our 2 tasks Task-1 and Task-2.

For Model 1 as illustrated in Fig.2, the sentence embeddings are computed for the input questions and one question is fed through a fully connected layer. We then apply dropouts and batch normalize the outputs. We then compute the dot product of Q1 and Q2' and feed the output of the dot product through a softmax layer to compute the final probabilities.

Model 2: Cross-Attention with Concatenation using USE

For Model 2 as illustrated in Fig.3, both the questions are concatenated together and an embedding is generated using USE. The embedding is fed through a dense fully connected network. We then apply dropouts and batch normalize after each dense layer output, and finally feed the output to a sigmoid activation function to predict whether the question is a duplicate.

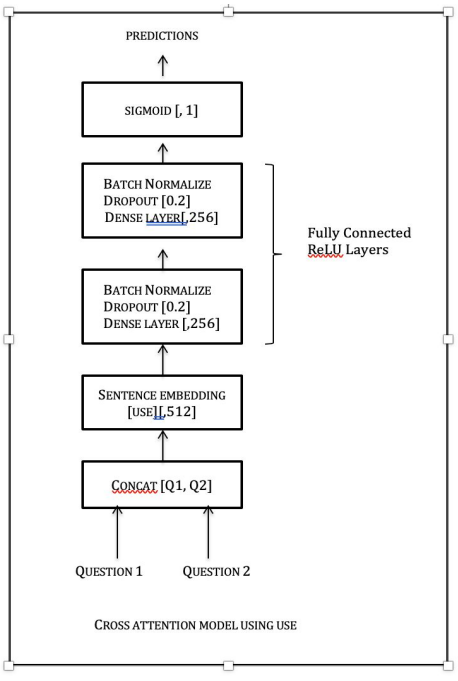


Fig.3 Cross attention model using USE for Task-1

Model 3: Cross-Attention with Concatenation using BERT

In Model 3 as illustrated in Fig.4, we generate input-IDs for all tokens in the sentence, corresponding masks, and segment IDs. We append the CLS and SEP tokens to each question pair. This is fed through the BERT layer, with the pooled output of the BERT layer being fed to a sigmoid activation function. We use a lower learning rate of 5e-5 for the Adam optimizer. The BERT weights were not frozen and no additional dense layers on top of the BERT layer were added.

Contrasting the cross-attention model architectures between BERT and USE, the BERT based model is significantly simpler and there are no additional denser layers added on top as compared to the USE based model where we had to add a fully connected network to generate good model performance.

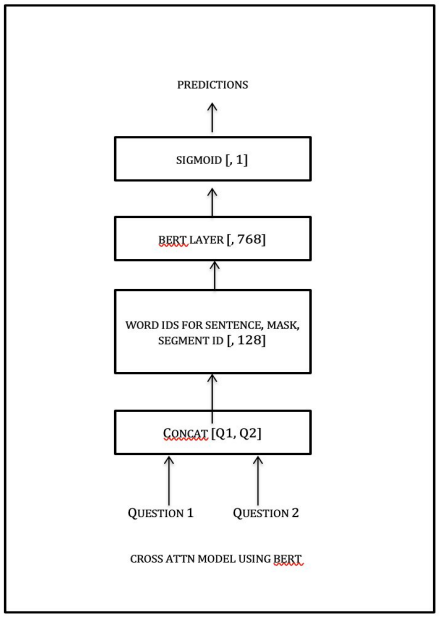


Fig. 4 Cross Attention model using BERT Task-1

Baseline

For the baseline model, we use Cosine Similarity between the sentence embeddings of the question pairs. The sentence embeddings are computed using USE. A logistic regression is trained to then compute a baseline score based on the classification of the binary labels. We used the USE based cosine similarity baseline given that it has provided reasonable results in many prior experiments.

As mentioned, the model architectures are applicable to both the tasks and our experiments involved training and validating both the tasks using identical model architectures with few exceptions. For the different tasks, we used different models of USE, namely a more generic model for the sentence similarity task and a QA specific model for the best response task.

Model Configuration

Model configuration and hyperparameters are set based on our experiments with the Stack Overflow dataset. The model parameters used are different for each task. For the question similarity task, we use SGD with a batch size of 256 and a learning rate of 0.01. There are 152K total training steps. We use the USE sentence embedding size of 512 for our 2 tasks based on USE and 768 for our BERT-based model. We also lower the learning rate for our BERT-based model to 5e-5. The data had an 80/20 split applied for our training and testing sets. The training data was further split 90/10 into training and validation sets. We use dropouts of 0.2 and batch normalization for all the dense layer outputs. Binary cross-entropy was used as the loss function for all our neural models.

For the best response task, there are 20K training steps and a batch size of 50. The reduced batch size is due to memory constraints on the cloud instance. We also reduce the training steps to improve our training times. Other parameters remain identical to the question similarity task.

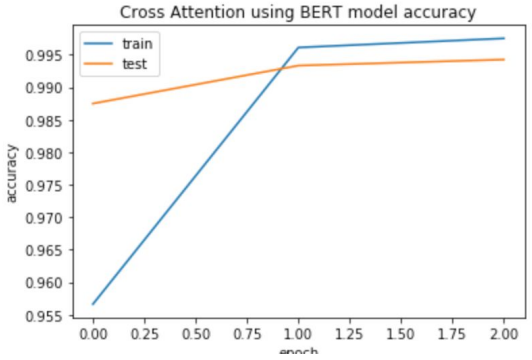
Model Results

Table-3 and Table-4 below highlight the results for our model performance across both tasks.

Question Similarity Task

#	Model	Accuracy
0	Cosine Similarity using USE	64.87
1	Two Tower using USE	94.61
2	Cross Attention with USE	94.90
3	Cross Attention with BERT	99.48

Table 3. Model Performance for Task-1

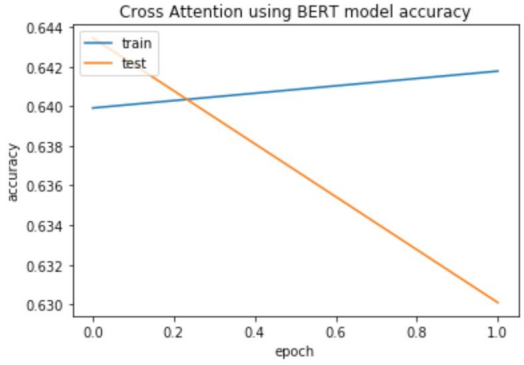


Graph-1: Model-3 Accuracy for Task-1

Best Response Prediction

#	Model	Accuracy
0	Cosine Similarity using USE	59.04
1	Two Tower using USE	61.02
2	Cross Attention with USE	61.83
3	Cross Attention with BERT	64.90

Table 4. Model Performance for Task-2



Graph-2: Model-3 Accuracy for Task-2

Model Analysis

We evaluate the four models described above on the question pairs training data for semantic similarity. Test accuracy was used as the main evaluation metric for both tasks. We measure whether the model is able to correctly classify a question as duplicate or not, given a pair of questions. After tuning the models and the hyperparameters appropriately, we see a progressive increase in our accuracy from our baseline model of Cosine Similarity. Our models using USE embeddings perform reasonably well. Our best performing model is the cross attention model trained using BERT embeddings. The table below shows the test accuracy metric for the question pairs dataset for 100K question pairs split into 80% train and 20% test sets.

We evaluate similar models for the best response task that involves predicting the correct “accepted answer” for a given question. The positive answers are accepted answers by the asker of the question. We ran this on 100K question answer pairs tagged for Python. There are a mean of 3.8 answers per question, out of which 1 was chosen as the accepted answer by the asker. The number of answers per question range from 2 to 63 answers per question. We removed the code snippets from the answers to reduce the record size.

The models did not fare as well for this task, as shown in table 5.

	Precision	Recall
Positive Class	0.47	0.23
Negative Class	0.66	0.86

Table 5. Precision and Recall using Cross Atten. With USE for Task-2

Investigating the data further, the shortcomings of the model can be in part attributed to the following challenges:

1. The model was not able to perform particularly well when answers contain large amounts of technical jargon or code snippets
2. Accepted answers that are very short and have links to external URLs or documents are challenging to classify, as they lack necessary context
3. The hard negatives in the dataset are adding some noise to the model
4. There is a human-element of why a particular answer is accepted out of a set of answers. This aspect is extremely challenging to capture in the model.

The example below illustrates an example where our model misclassified a response due to #2 above

Question: "How do you extract the word from 'what does [word] mean' in a python string?"

Best Response: "As says https://docs.python.org/2/library/string.html#string.split"

Model Prediction: "The easy way is to use a regular expression such as.."

The best response in the example was simply a link to an authoritative resource, namely the official Python Documentation. Removing the punctuation turns the URL into a nonsensical collection of letters. Our model predicted the other response, which, while not much more detailed, did provide some related terms to the question, like "regular expression". It should be noted that both of the responses in their original form contained code excerpts, which were excluded from our models.

Conclusion

In this paper, we apply semantic similarity and response prediction techniques to programming Q&A. We observe that the question similarity models perform exceptionally well on sentence level similarity and our cross-attention model using USE and BERT provide encouraging results. Our second task of response prediction delivers modest results and we outline some of the challenges of modeling the best response identification. Our future research will explore applying domain adaptation and transfer learning frameworks to improve the response prediction model and explore additional tasks required for a conversation agent for programming Q&A.

Acknowledgements

We would like to acknowledge the W266 faculty, particularly Daniel Cer for his exceptional feedback and mentorship and Mike Tamir for his guidance on this work.

References

1. Bowman, Samuel R., et al. "A large annotated corpus for learning natural language inference." *arXiv preprint arXiv:1508.05326* (2015).
2. Cer, Daniel, et al. "Universal sentence encoder." *arXiv preprint arXiv:1803.11175* (2018).
3. Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
4. Henderson, Matthew, et al. "Efficient natural language response suggestion for smart reply." *arXiv preprint arXiv:1705.00652* (2017).
5. Yang, Yinfei, et al. "Learning semantic textual similarity from conversations." *arXiv preprint arXiv:1804.07754* (2018).
6. Adiwardana, Daniel, et al. "Towards a human-like open-domain chatbot." *arXiv preprint arXiv:2001.09977* (2020).
7. Fast, Ethan, et al. "Iris: A conversational agent for complex tasks." *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018.

8. Ma, Xiaofei, et al. "Domain Adaptation with BERT-based Domain Classification and Data Selection." *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*. 2019.
9. Dehghani, Mostafa, et al. "Learning to transform, combine, and reason in open-domain question answering." *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 2019.
10. Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.
11. https://console.cloud.google.com/marketplace/details/stack-exchange/stack-overflow
12. https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313