Finding Answers to Non-factoid Questions Using a Recursive Neural Network

Deepak Merugu, Reed Nightingale, Imran Thobani

Abstract

Retrieving answers to complex questions is an unsolved problem in computing. Surdeanu et al. have shown that by identifying key features in the sentences in both a question and set of candidate answers, a combination of discriminative learning and class-conditional generative algorithms can correctly identify the best answer (as determined by a human) with a baseline precision of 0.41 and mean reciprocal rank (MRR) of 0.56 [1]. We have produced a prototype, alternative model by using an adaptation of the recursive autoencoder (RAE) model presented by Socher et al[2]. While our model does not currently achieve a testing precision or MRR better than Surdeanu's model, with some changes to the approach to training the model, it may still be possible to match or exceed Surdeanu's results.

Introduction

In recent years there has been a significant growth in user-generated content on the web. Specifically, there is a huge collection of questions and answers, written and rated by users, on social question answering websites such as Yahoo! Answers and Quora. This has led to interest in building a question answering system using machine learning techniques. IBM Watson and Apple's Siri are examples of such systems that have received a lot of attention.

Recent work by Surdeanu et. al.[1] shows that certain linguistically motivated features improve the performance of the ranking module of a complete question answering system. In a related paper, Richard et. al.[2] use recursive auto-encoders (RAEs) for the task of paraphrase detection. Building on this work, we propose to use a similar model with different pooling techniques to learn similarities between questions and answers and thus, rank answers from a given pool. At a high-level, our model represents each sentence (from a question or an answer) as a tree of words (leaf nodes) and phrases (parent nodes). Words and phrases are represented as n-dimensional vectors in a vector space, and phrases are generated by recursively applying a neural network to child nodes. We use a similarity matrix combined with a pooling technique to compute a fixed number of similarity features between the sentences, which are then used to classify a question-answer sentence pair as correct or incorrect.

Approach

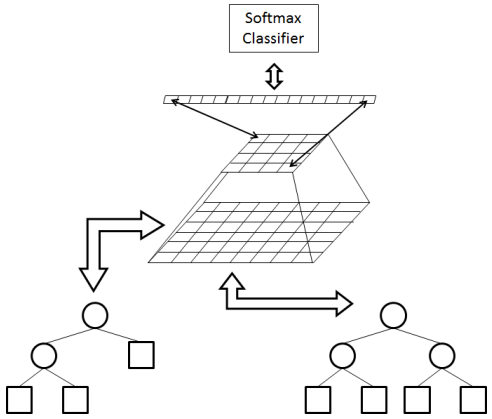
The RAE model presented by Socher et al was originally designed to detect paraphrased sentences. Because a good answer to a question will often repeat key words and phrases from the question, we suspect that the same general model may be trained to match questions and answers.

The model has three main units. At the lowest level, a RAE parses and processes sentences into trees (representing the sentences' syntactic structures). This piece of the model uses vector representations of words and phrases, and is designed to train semantically and syntactically similar words and phrases to have similar vector representations.

Interfacing with the RAE is a sentence comparing and pooling layer. In our tested implementation, this layer compares two sentences by calculating the squared Euclidean distance between each word, phrase, or word-phrase pair of vectors to generate a similarity matrix.

Because the size of the similarity matrix varies based on the length of the sentences provided to it, there is a pooling layer on top of the similarity matrix that averages roughly equal sections of the similarity matrix to produce a fixed size pooled similarity matrix.

At the top level, a softmax classifier uses the pooled similarity matrix to determine the model's classification of the two sentences as a good or bad question/answer sentence pair. To do this, the softmax classifier is trained on a data set that is labeled as 1 for sentence pairs in a good answer for a question and 0 for sentence pairs in a bad answer for a question.



Any discrepancy is error that is back propagated from the softmax classifier to the pooled similarity matrix, down to the source similarity matrix, and finally to the sentence trees and word and phrase vectors. Stochastic gradient decent (SGD) is then used to minimize the cost function and cluster the word vectors. Thus the two ends of the system can simultaneously be trained in a single pipeline.

To test our model, we used a Terrier IR BM25 algorithm[3] to generate a list of 15 candidate answers for each question selected for relevance. For each trained model, we tested all of the questions the model was trained on, and also tested the model against a fixed set of 165 questions used exclusively for testing. Given a question, we used the learned word vectors and softmax classifier weights to determine a classification probability for each answer, which was translated into ranks with the highest softmax classification answer receiving a rank of 1, the lowest 15, and any ties broken by randomly ranking the answers within the appropriate rank-range. From the rankings, it is straightforward to calculate the precision and MRR for the model.

Our code was developed in Java on top of a suite of code originally written by Richard Socher, and translated to Java by Jean Feng. After determining the requirements for our model, we had to modify several portions of the original framework in order to allow our code to interface with the existing framework.

Training and Testing Results and Analysis

We designed unit testers to check the correctness of each individual component in the system. After each component was successfully tested, we did a numerical computation of the gradient and verified that the back-propagation phase indeed returns the true gradient. In using SGD, we also checked that the cost was decreasing with every update of the parameters for different learning rates.

We now present some results and observations for different test cases. Figure 1 shows precision and MRR for a training data set of 5 question-answer lists (about 15 question-answer pairs). We can observe that the model fits the training sets almost perfectly after about 300 iterations. The initial weight matrices have been randomly initialized differently for the case where we did not use any regularization of the parameters and the case where we did. This shows that the model does indeed learn to rank correctly, at least for a tiny data set.

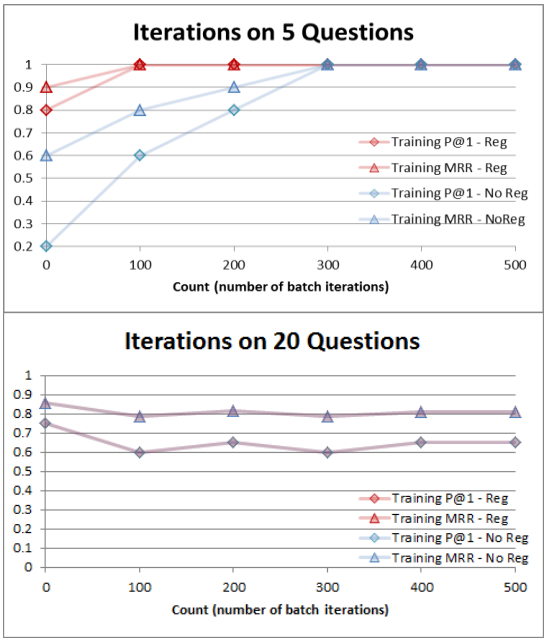


Figure 2 shows the learning curve on a training set of size 20 questions (about 60 question-answer pairs) with and without the regularization of the parameters. In both cases, the parameters have been initialized to the same values. It is interesting to observe very similar performance in both cases - showing that the regularization is in fact not hurting the model performance.

Figure 3 shows the learning curve on a training set of size 100 questions (about 300 question-answer pairs) for about 600 iterations. The learning curve does not show any reasonable improvement in the performance.

Figure 1 and 2

Figure 4 shows the performance on a training set of size 2500 (6,730 question-answer pairs) questions comparing it to the case where we only test on the original answers for a question, which is variable.

Figure 5 shows the performance with the size of the training set. While we see a slight improvement in the performance as the training size increases to 10,000 (32,531 question-answer pairs) questions, there is also a dip for a training size of 14,500 (50,674 question-answer pairs) questions.

Figure 6 shows a sample set of pooled similarity matrices between several sentence pairs. Darker blues indicate closer vectors, while dark red indicates large differences. Medium values are shades of yellow and green. Fig.6a,b,c compare the sentence "Does China view North Korea as a potential threat to its security?" to one random sentence (a) and two sentences identified by the IR (b,c). We see that there is a greater amount of bluer regions in b and c than in a. Fig.6d,e show the similarity matrices for two best answers and their respective questions. We note that d appears rather dissimilar, and e has regions of extreme similarity but also several dark red regions.

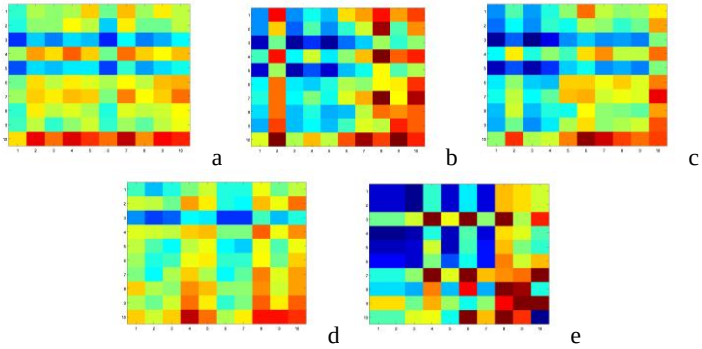


Fig. 6(a-e). Sample similarity matrices for different question answer pairs.

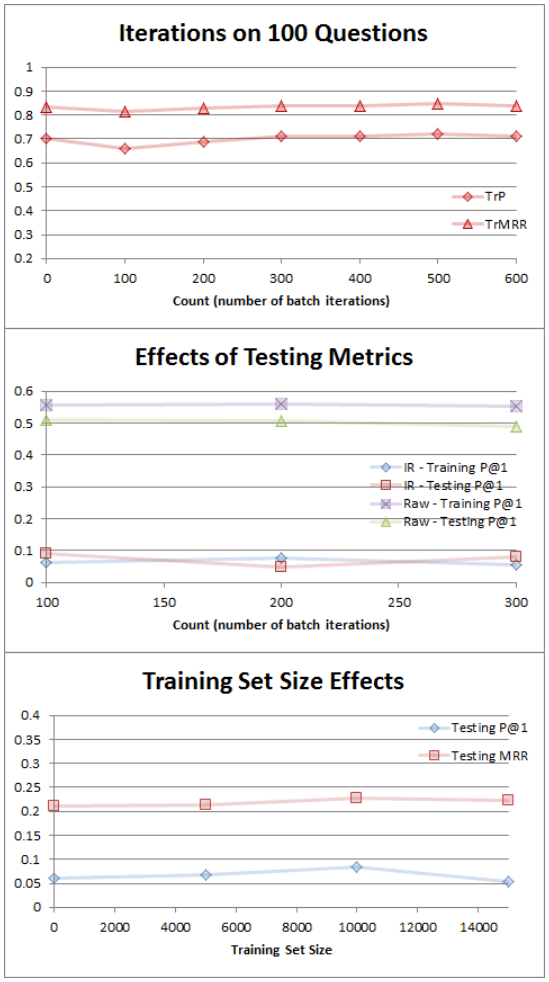


Figure 3, 4, and 5

We also tried training the model over 5000 question-answer pairs using Collobert and Weston's word vectors as initial values for the model, but found no improvement in over 80 batch iterations. We suspect that this could still be an effect improvement to the model with a superior optimization method such as L-BFGS because the pre-trained word vectors we used were not pre-trained sufficiently based on nearest neighbors analysis. We tried to optimize our model using L-BFGS, but the optimizer we used did not appear to update the weights of the word vectors. A future goal would be to include a superior optimizer.

Conclusion and Future Work

There are several variables left to be tested, but three in particular may have significant impact on the model's performance. The first of these is the training algorithm. Our results all use SGD, which did not demonstrate an appreciable learning curve for data sets larger than five questions; however, it is possible that an optimizer such as L-BFGS would have led to better results. While we have implemented and unit tested a series of pooling styles, the only one used to obtain these results so far is the average pooling. It may be

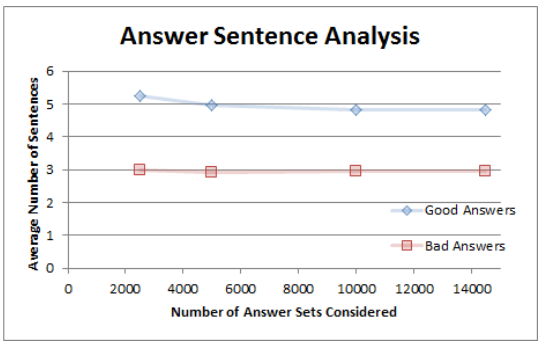


Figure 7

that one of the other three types (minimum, minimum Gaussian weighted, and exponentially weighted) produces consistently better rankings and/or precision. Finally, due to the relatively low number of training iterations, questions trained over, and word vector size, it is difficult to determine how our model will compare to Surdeanu's when given more ample iterations to learn word features.

To improve the performance of the model, we might consider adding additional features. For example, TFIDF, number of sentences in answers to question, as seen in figure 7, or named entity recognition.

Acknowledgements

We conducted this research within the Deep Search Natural Language Processing group at Stanford University, under the supervision of Richard Socher, in collaboration with Rahul Pandey and Arun Prasad, using a framework of code developed by Jean Feng.

References

[1]Surdeanu, Mihai, Massimiliano Ciaramita, Hugo Zaragoza. "Learning to Rank Answers to Non-Factoid Questions from Web Collections." Computational Linguistics, Vol. 37, No. 2. June 2011.

[2]Socher, Richard, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, Christopher D. Manning. "Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection." Advances in Neural Information Processing Systems 24. 2011.

[3] Terrier v3.5. 16-06-2011. http://terrier.org/