Fine-Grained Question Similarity Detection

Priyanka Sekhar
Stanford University
psekhar@stanford.edu

Ruben Mayer
Stanford University
rmayer99@stanford.edu

Konstantine Buhler
Stanford University
buhler@stanford.edu

Abstract

The task of fine-grained question similarity detection has meaningful applications in many automated conversations. Our project investigates the effectiveness of SVMs and Shallow Neural Networks for this task. Both the Neural Network and the SVM outperformed our baseline by significant margins. The ablation test revealed that word overlap features were effective measures of similarity, and our optimal feature set realized 83.3% accuracy on the test set, outperforming our Neural Network. Further tuning and refinement would likely enhance Neural Network performance, and future studies will address the adaptability of this work to larger and more complex datasets.

Introduction

Question Answering is a major field of study within NLP and has interesting applications for chatbots. Sentence similarity detection allows for better linguistic understanding and thus better Question Answering, and as a result, a large body of research is dedicated to combining many methods of NLP for effective machine conversation. The task of fine-grained question similarity detection has meaningful applications in many automated conversations, since effective question similarity detection would allow an agent to efficiently pull answers from a database of previously answered questions as well as augment an individual's ability to directly respond to many people.

1.1 Related Work

Foundational work addresses sentence similarity detection through word overlap and linguistic heuristics for paraphrase detection.[1] SVMs trained on the TREC dataset have also been shown to effectively group questions into broad topical categories.[2] Emerging work addresses the task through neural networks, with some research attempting to adapt the convolution neural network architecture traditionally used in image recognition to language processing.[3]

1.2 Proposed Solution

For our final project, we are investigating methods of determining if two questions are similar using paraphrase detection techniques. Based on the previous success of SVMs in language grouping tasks and the emerging promise of Neural Networks, we have decided to focus our investigation on these two models.

Methodology

The success of our two models was determined by comparison to a naïve baseline. Precision, recall, f1-scores, and accuracy were taken into consideration when determining usefulness for this task.

1.3 Baseline

As a simple baseline, we created a rules-based approach that uses a bag-of-words model with the Glove vector word representations created by the

[1] [1] Achananuparp et al.
[2] [10] Silva et al.
[3] [2] Kim et al.

Stanford NLP group.[4] Our algorithm creates a vector representation of each phrase by adding the Glove vectors that correspond to each word in that phrase. Subsequently, it compares different phrases by taking the cosine distance of the vectors corresponding to each phrase. Our rules based approach then distinguishes positive question pairs from negative question pairs by setting a threshold cosine distance that divides the two groups (where question pairs with a cosine distance smaller than 0.075 are deemed to have the same meaning, and question pairs with a cosine distance larger than 0.075 are deemed to have a different meaning). This approach led to an approximate accuracy of 56%.

	Positive	Negative
Precision	0.533	0.544
Recall	0.705	0.363
F1-Score	0.607	0.436

Table 1: Results of Baseline on Random Test Sample of 1000 question pairs

Note the low recall of negative examples means that this is a good baseline for comparison. The baseline does a poor job distinguishing different meanings from sentences with different structures, and our task here is to improve this performance. Additionally, because cosine distances detect syntactic similarity, these baseline results are good indicators that our dataset contains syntactically similar negative examples.

1.4 Models

The sklearn SVM implementation was used for all SVM evaluations. The RBF kernel was used for evaluation in all feature combinations. The kernel was chosen over the linear kernel due to non-linearity in our dataset. The RBF was chosen over other non-linear kernels from its superior performance in preliminary trials using randomly chosen feature subsets of varying length. From our complete SVM paraphrase dataset (described in more detail in Section 3), 33% of examples were randomly withheld from the training set for use as the test set. Coarse parameter tuning was performed using values ranging by an order of magnitude of 10 for C values between 0.1 and 10000 and for gamma values between 2^-8 and 2. These parameters were fixed for all feature combinations based on preliminary trials using randomly chosen feature subsets of varying length.

We began our exploration of Neural Networks by implementing the model used in the word-level entailment class on April 25, 2016. As part of this process, we leveraged the code developed in that bakeoff as well as the networks designed in shallow_neural_networks.py.[5] The first step in this process was formatting input.

Our initial attempt to format data relied on random vectors of length 50 for each of the two sentences. Results from this test were barely better than chance and on average resulted in an f1-score of 0.54. The next step was to swap out this encoding technique and instead use Doc2Vec feature representations. Doc2Vec has several useful characterizes that account for sentence structure and thus enable improved data representation. Each pair of doc2vec vectors u, v were formatting as follows: np.concatenate((u,v, u-v)). This format was used as the final input for the neural net and significantly increased accuracy relative to random vector representations.

We also experimented with other types of neural networks, including the TF (tensor-flow) equivalent that was made available in shallow_neural_networks.py, but improvements were negligible. In the end, we used this shallow neural network as our network of choice due to its ease of implementation and reliability.

Two parameters were the focus of neural network tuning: the number of hidden dimensions and eta. To determine the optimal number of hidden dimensions we systematically tested different values between 10 and 100 on a small dev set (~1000 examples) and compared resulting accuracies. If we hold all other parameters constant and vary hidden dimensions we find something like the following on dev data:

Hidden Dim	25	50	100	200
Dif	0.79	0.76	0.84	0.75
Sim	0.7	0.78	0.79	0.74
Avg	0.75	0.77	0.82	0.75

[4] [6] Stanford NLP

[5] [7] Potts

After testing several different values for these parameters, we chose a hidden dimension value of 100. Holding this value constant, we performed a similar analysis with various values for the step size, eta, and settled on an eta value of .05. Finally, we experimented with several different sizes for training and test sets. We concluded that 90% to training was fitting, as a network like the one we built benefits from more data. In the end, we had a total of 5,000 positive examples and 5,000 negative examples for use in our training and test sets.

2 Data

We used a dataset of questions that was previously scraped from WikiAnswers.[6] WikiAnswers is a webpage where users can ask questions about any topic that they like, and they are answered by the public. When several users ask the same question, editors from the site can merge them so that they receive the same answer. The questions are all saved, and identified as having the same meaning. Hence, the dataset is composed of many sets of question pairs that have the same semantic meaning, but have a different lexical form. A few examples of our positive dataset include:

"How many cups are there in a gallon?" – "A gallon contains how many cups?"

"How many Olympic medals has Michael Phelps won?" – "How many Olympic medals does Michael Phelps have?"

"What is a habitat of a rhino?" - *"What natural habitat do the rhino have?"*

2.1 Negative Dataset

Our negative data set was constructed using n-grams and is composed of pairs of questions that fulfill three criteria: First, the sentences that compose each pair have a different semantic meaning to each other; second, the negative pairs have a similar lexical form to the positive samples in the dataset (i.e. they have similar length and use similar types of words); and finally, while the pairs are composed by sentences with a strictly different semantic meaning, the sentences' meaning still has a high degree of similarity. This ensures that our negative dataset is composed of pairs that resemble the real-world paraphrase problem, where the pairs that are tested come from a similar context, and therefore have a high resemblance towards each other.

We constructed our negative dataset by using sentences taken out of the WikiAnswers Paraphrase Dataset, and comparing them to each other using a simple word-overlap model. In order to be included into the negative dataset, two sentences had to fulfill the following criteria:

- The sentences had more than three overlapping words (excluding the common words "be", "the", "of", "do", "a", "in", "for", and "to)

- The set of overlapping words was smaller than the total amount of words in the shorter of the two sentences minus two (this ensured that the sentences were not identical)

- The difference between both sentences' length was smaller than four (this prevented sentences from having a large word overlap simply because of the size of one of them)

Examples of negative pairs include:

"How much do 4 AA batteries cost in USD?" - *"How much does a custom Bugatti cost?"*

"How many centimeters equal 15.6 meters?" - *"How many kilometer equal 250 meters"*

"What is the difference between the French and the american Revolution 's goal?" - *"What is the main difference between c3 and c4 carbon fixation?"*

In total, our dataset was composed of 5,000 positive pairs, and 10,000 negative pairs for training the Neural Network. We had more negative pairs to account for the lack of precision in our negative dataset given our construction method, which causes a certain degree of variation in the semantic distance between the pairs in the dataset. Because Glove representations were used

[6] [14] Wikipedia Dataset

in SVM feature creation, we limited the dataset for SVM training and testing to sentences that could be completely represented with the given pre-trained Glove vectors. This dataset contained 4246 positive question pairs and 4724 negative question pairs.

3 Experimental Setup

A total of 9 features[7] were used in an ablation test for the SVM using the Glove compatible dataset. Training and testing were performed using the sklearn implementation on all feature subsets.The Neural Network hidden dimensions remained fixed at 100 for subsequent trials.

3.1 SVM Features

0 - d2vcos: Cosine distance between Doc2Vec[8] sentence representations. Paragraph to Vector is a method for unsupervised semantic analysis of texts with an arbitrary length that is based off of the bag of words approach. However, unlike bag of words models, the Paragraph to Vector algorithm takes word ordering into account when constructing vectors. This method was used to generate doc2vec representations of all test and train examples.

1 - glovecos: Cosine distance between the Glove Vector representations of the two sentences. Implementation details can be seen in our GitHub.[9]

2 - unigram: Unigram set overlap. The integer size of the intersection between the set of individual words in each sentence.

3 - jaccard-uni: Jaccard Coefficient between set of overlapping unigrams. The size of the intersection of the unigram set divided by the size of its union.

5 - qword: Presence of the same question word (case insensitive) in each sentence. The question words used are: “who,” “what,” “where,” “when,” “why,” and “how.”

[7] A 10th feature, Euclidean distance, was implemented but not tested. Future studies may include this feature. Index 4 is thus not listed in the SVM feature descriptions.
[8] [4] Distributed Representations of Sentences
[9] https://github.com/konstantine33/CS224uSentenceSimilarity.git

6 - bigram: Bigram set overlap. The integer size of the intersection between the set of individual bigrams in each sentence.

7 - jaccard-bi: Jaccard Coefficient between set of overlapping bigrams. The size of the intersection of the bigram set divided by the size of its union.

8 - ner: Named Entity presence**.** Indicator feature that takes a value of 1 when only one of the sentences in the question pair contains a Named Entity. Named Entities were parsed using Stanford CoreNLP NER.[10] The set of Named Entities included are: Location, Person, Organization, Money, Percent, Date, Time.

9 - root-dist: Cosine distance between roots of the dependency parse. Dependency trees were obtained using the Stanford Parser.[11]

10 - tree-depth-def: Magnitude of the difference in dependency tree depth obtained using the Stanford Parser. The depth was taken as the maximum depth for each sentence tree.

3.2 SVM Evaluation

For a complete list of precision, recall and f1 scores across all combinations of features, refer to our GitHub repository.[12] The best and worst performers (determined by overall accuracy) are listed here for reference. Feature set numbers correspond to features described in Section 4.1.

Feature Set	**Accuracy**
1, 2, 3, 5, 7, 8	0.833
1, 2, 3, 5, 6, 7, 8	0.831
1, 2, 3, 5, 7, 10	0.831
1, 2, 3, 5, 7, 8, 10	0.830
0, 1, 3, 5, 6, 7	0.829
0, 1, 2, 3, 5	0.829
0, 1, 2, 3, 5, 8	0.829
1, 2, 3, 5, 7	0.828
0, 1, 2, 3, 5, 7	0.828

Table 2: Top Performing Feature Subsets

[10] http://nlp.stanford.edu/software/CRF-NER.html
[11] http://nlp.stanford.edu/software/lex-parser.shtml
[12] https://github.com/konstantine33/CS224uSentenceSimilarity.git

Feature Set	**Accuracy**
10	0.534
0	0.551
8	0.559
9	0.561
0, 8	0.563
0, 9	0.571
0, 10	0.572
8, 10	0.572
9, 10	0.578

Table 3: Bottom Performing Feature Subsets

	Precision	Recall	F1-Score
Dif_Neg	0.815	0.885	0.848
Sim_Pos	0.858	0.776	0.815
avg / total	0.835	0.833	0.832

Table 4: Precision, Recall, and F1-Score for top performing Feature Set (1, 2, 3, 5, 7, 8)

The SVM was able to achieve approximately 83% test accuracy, which far outperforms our baseline. Features 1, 2, 3, and 5, which are essentially various forms of meaningful word overlap representation, appear in each of top performing feature sets. Feature combinations that were made exclusively from features 0, 8, 9 and 10 filled the bottom performance tier.

Below are comparisons to our top SVM results to results from reference literature and related works.[13]

Model	**Accuracy**
SVM	0.833
Xu et al.	0.915
Achananuparp et al. (word overlap features)	0.819

Table 5: Comparison to Reference Work

The distribution of performance across features makes sense given the low complexity of our dataset. Because all of our questions are of similar length and many contain similar key ideas, features such as Named Entity Presence (Feature 8) and root-dist (Feature 10) do not introduce meaningful reparability in the data in isolation.

3.3 SVM Feature Comparison and Observations

Feature set (1, 5) alone produced and accuracy of 0.815, while feature set (3, 5) resulted in a comparable accuracy of 0.805. This strong performance indicates that a combination of these features is the primary cause of significant separation in the data. In fact, features sets that contain features 1, 3, and 5 tend to have accuracies in the high 70s and low 80s.

The effectiveness of these feature combinations makes sense because features 1 and 3 tend to be strong indicators of positive examples while feature 5 is an extremely strong identifier of negative examples. In isolation, feature 5 does a rather poor job of identifying positive question pairs, with a positive recall of 0.552, which is less than the baseline and little better than random. However, it is a strong indicator of a negative example and thus serves as an effective counterbalance to word overlap features.

Feature 9 did little to enhance SVM performance. For example, precision, recall, and f1-scores for feature sets (3) and (3, 9) were almost identical. This is likely an indication that the dependency parse roots were similar for many of our negative examples. A more effective feature for root parsing might instead be an indicator function for determining when the two questions contain exactly the same root rather than a similar one.

Feature 10 occasionally enhanced feature set performance. In isolation, its performance was essentially random, likely because this dataset contained questions of similar length. As a result, relevant question complexity differences were likely better captured in the jaccard-uni feature, which takes into account question size. However, this feature could provide more nuance for higher complexity datasets.

Feature 8 also appeared in both the best and worst performing feature sets. This is likely because the presence of a named entity is not a very strong indication of a positive pair, but a difference in named entities is a strong indicator of

[13] Note that our datasets are different from the comparison datasets (Chinese corpus and TREC9 respectively). However, the comparison serves to demonstrate that our results at this stage are comparable to current research work.

a negative example. Thus, this feature enhances the performance of sets that tend to have a bias for labeling two questions as positive by adding nuance to negative data representation.

The doc2vec representations appear in both the bottom and the top performers, likely because they are more suited to long questions in complex datasets and only slightly enhance the models ability to recognize positive examples. While these vectors were critical in enhancing the performance of the neural network, their effect on SVM performance was minimal.

Both bigram and unigram functions had high recall (~0.9 each) for negative examples, indicating a strong bias in that direction. The jaccard measures for each of these sets provided more balanced precision and recall estimates.

Overall, features 1, 2, and 3 seemed to be the strongest indicators that two questions were similar. On the other hand, features 5 and 8 tended to be the best indicators of negative questions in this dataset. As a result, feature sets that contained some combination of features from the strong positive indicators (1, 2, 3) and the strong negative indicators (5, 8) tended to perform very well on this task.

3.4 Neural Network Evaluation

We ran the Neural Network five times on 10,000 instances each time (90% train and 10% test), and attained the following results from our network analysis.

Train	Precision	Recall	F1-Score
Dif_Neg	0.72	0.74	0.73
Sim_Pos	0.74	0.71	0.72
avg / total	0.73	0.73	0.73

Test	Precision	Recall	F1-Score
Dif_Neg	0.68	0.70	0.71
Sim_Pos	0.69	0.66	0.67
avg / total	0.69	0.69	0.69

Table 6: Train and Test Performance of Shallow Neural Network

On average, the Neural Network had lower precision, recall, and f1-scores on the test set than our top SVM feature set, with a top accuracy of 0.74 on the test set.

3.5 Direct Comparison Summary

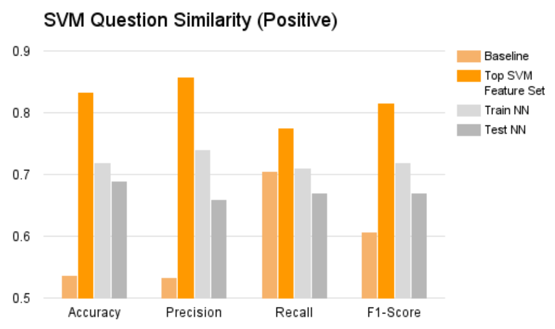

Chart 1: Comparison of Baseline, SVM, and NN for Positive Question Classification

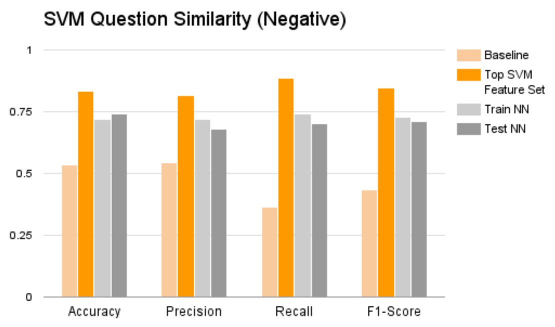

Chart 2: Comparison of Baseline, SVM, and NN for Negative Question Classification

As evidenced above, the SVM significantly outperforms the Neural Network across metrics for both negative and positive example classification. The Neural Net also significantly outperforms the baseline, which indicates promising future work in Neural Net refinement.

Discussion

Overall, our models performed as expected and comparably to current work, and we have a solid foundation for continued research and refinement. The SVM was able to realize higher performance than the Neural Network for several of the different feature subsets. This performance difference is likely because the hand-curated SVM features account for well-known linguistic indicators that the Neural Network could not determine from our vector representations. For example, the use of bigram overlap and Named

Entity recognition is not easily intuited without human input. Furthermore, parameter tuning was based on our own coarse approximations from a subset of runs. Nuanced parameter tuning would be required to increase Neural Network performance relative to the SVM.

3.6 Dataset Considerations

We independently generated our own negative data; however one concern with our method is that the negative labels are not guaranteed to be truly negative. While upon cursory human inspection, the negative labels do generally appear to be non-paraphrases, the n-gram method itself does not definitively guard against accidental paraphrase creation. This potentially limits the extent to which we can optimize performance.

Our dataset is also low complexity, meaning sentences tend to be short and absent of complex linguistic features. Although our models perform reasonably well on this dataset, the data itself does not provide much information about how our models generalize to more complex sentences.

While these considerations may limit peak performance of both the Neural Net and the SVM, the ablation test reveals relevant insights on the types of features that introduce separability in question similarity data and generally enhance prediction for this task.

Future Work

While our preliminary investigation revealed informative trends, future studies could increase absolute accuracy through more precise SVM parameter tuning. The current set of gamma and C values were held constant for each feature set, and continued studies would refine these parameters for different subsets. This might enhance the performance of sets that tend to have a high bias. Different train/test splits with different data sizes would also increase the robustness of our results. Additionally, the work of Achananuparp et al. suggests that the incorporation of semantic similarity features and WordNet-based measures will further increase our peak SVM performance.[14] Our current dataset was limited by the pre-trained Glove vector embeddings, and further studies would investigate suitable replacements for use on larger test datasets.

Another immediate next step would be a thorough analysis of accuracy clusters for sets containing different features. The observations provided here lay the foundation for a more quantitative representation of correlations between features and precision, recall, and accuracy.

We only tested the results of our model on a single dataset. However, the TREC9 dataset would allow a more direct comparisons with existing work and provide validation for our model on low-complexity question paraphrase detection tasks.[15] Additionally, the MSRP and RTE3 datasets as used by Achananuparp et al. could provide insight into how our model performs on high complexity questions. As a result, future work would incorporate a greater diversity in data curation.

Neural Network performance could be enhanced through codified parameter tuning and iterations on effective data formats. For example, it is possible that we could have done better by using a representation other than doc2Vec. Characteristics such as the length of the document could have also been considered. Future work in relation to neural nets would thus likely investigate both different vector representations and varying vector association calculations, to expand upon the results obtained using vector concatenation and difference.

Finally, we saw some promising research about the use of CNNs in this field. This field of research is relatively undeveloped, and thus another potential continuation of this project could investigate the creation of new lenses to convolve sentence-similarity data for effective use with a CNN.

Conclusion

Our baseline, Neural Network, and SVM all significantly outperformed random chance for the sentence similarity task. On average, the Neural Network had lower precision, recall, and f1-scores on the test set than our top SVM feature set, which attained a notable average F1 score of 0.832, putting it on par with other research. The Neural

[14] [1] Achananuparp

[15] http://trec.nist.gov/data/qa/t9_qadata.html

Network also significantly outperformed the baseline, although with a lower average F1 of .69. We believe that with further research, some of which was outlined in “Future Work,” fine-grained question similarity detection will continue to advance steadily.

Question Answering is a significant field of study within NLP and sentence similarity detection is fundamental to the success of this task. Our hope is that our continued work in this field will contribute to advancements with intelligent virtual assistants and chatbots. Ultimately, as techniques become increasingly reliable and robust, the ability for a computer to efficiently address human questions through techniques such as fine-grained question grouping will have far-reaching effects in both academic and commercial spheres.

Acknowledgments

We would like to thank Bill MacCartney, Chris Potts, and Matt Lamm for their help in guiding the project’s vision and for their suggestions throughout the course of this quarter.

Citations

[1] Achananuparp, Palakorn, Xiaohua Hu, and Xiajiong Shen. "The evaluation of sentence similarity measures." Data warehousing and knowledge discovery. Springer Berlin Heidelberg, 2008. 305-316.

[2] Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).

[3] Ko, Youngjoong, and Jungyun Seo. "Automatic text categorization by unsupervised learning." Proceedings of the 18th conference on Computational linguistics-Volume 1. Association for Computational Linguistics, 2000.

[4] Le, Quoc V., and Tomas Mikolov. "Distributed representations of sentences and documents." arXiv preprint arXiv:1405.4053 (2014).

[5] Li, Jiwei, et al. "A Persona-Based Neural Conversation Model." arXiv preprint arXiv:1603.06155 (2016).

[6] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.

[7] Potts, Chris. “Word-level entailment with neural networks” Bake-off and accompanying material.

[8] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

http://turing.cs.washington.edu/papers/acl-2013-fader.pdf A Classification of Questions using SVM and Semantic Similarity Analysis

[9] Shang, Lifeng, Zhengdong Lu, and Hang Li. "Neural responding machine for short-text conversation." arXiv preprint arXiv:1503.02364 (2015).

[10] Silva, L. Coheur, A. Mendes, A. Wichert. 2011. From symbolic to sub-symbolic information in question classification. Artificial Intelligence Review, 35(2):137–154.

[11] Shawar, Bayan Abu, and Eric Atwell. "Different measurements metrics to evaluate a chatbot system." Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies. Association for Computational Linguistics, 2007.

[12] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.

[13] Vinyals, Oriol, and Quoc Le. "A neural conversational model." arXiv preprint arXiv:1506.05869 (2015).

[14] Wikipedia. Close Paraphrasing dataset. https://en.wikipedia.org/wiki/Wikipedia:Close_paraphrasing.

[15] Zheng, Suncong, et al. "A Novel Hierarchical Convolutional Neural Network for Question Answering over Paragraphs." 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). Vol. 1. IEEE, 2015.